Concepts et méthodes pour le développement d'environnements de simulation intelligents. Application au bâtiment

Souheil Soubra

▶ **To cite this version:**

Souheil Soubra. Concepts et méthodes pour le développement d'environnements de simulation intelligents. Application au bâtiment. Modélisation et simulation. Ecole nationale des ponts et chaussées - ENPC PARIS / MARNE LA VALLEE, 1992. Français. NNT : . pastel-00574043

HAL Id: pastel-00574043

https://pastel.archives-ouvertes.fr/pastel-00574043

Submitted on 7 Mar 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° 46 [illegible]

THESE

Présentée à

L'ECOLE NATIONALE DES PONTS ET CHAUSSEES

en vue de l'obtention du diplôme de

Doctorat de l'ENPC

Spécialité : Sciences et Techniques du Bâtiment

Par

SOUBRA Souheil

CONCEPTS ET METHODES POUR LE DEVELOPPEMENT D'ENVIRONNEMENTS DE SIMULATION INTELLIGENTS

APPLICATION AU BATIMENT

Soutenue le 08 Juillet 1992
devant le Jury composé de

MM.	Jean BRAU	Président
	Alain CORDIER	Rapporteur
	Jean-Claude MANGIN	Rapporteur
	Bertrand DELCAMBRE	Directeur de thèse
	Roland FAUCONNIER	Examinateur
	Roger PELLETRET	Examinateur
	Claude-Alain ROULET	Examinateur

79772

NS 16494
(4)

A Michaël-Bilal,
A ma famille,
A mes parents.

ECOLE NATIONALE DES PONTS-ET-CHAUSSÉES
CENTRE PÉDAGOGIQUE DE DOCUMENTATION ET DE COMMUNICATION

E.N.P.C.
DOC09101

32

AVANT-PROPOS

Les travaux qui font l'objet de ce mémoire ont été réalisés à l'établissement de Sophia Antipolis du CSTB (Centre Scientifique et Technique du Bâtiment). Mes remerciements vont à Monsieur Michel RUBINSTEIN, Chef de l'établissement de Sophia Antipolis, pour avoir mis à ma disposition les moyens nécessaires pour mener à bien cette étude ainsi qu'à Monsieur Bertrand DELCAMBRE, Chef du Service Informatique et Bâtiment, pour avoir accepté de diriger ce travail de recherche.

Je tiens à remercier tout particulièrement Monsieur Jacques RILLING, Directeur Scientifique du CSTB, sans l'aide duquel ce travail n'aurait jamais démarré ni abouti. Qu'il trouve ici l'expression de ma profonde gratitude.

J'adresse mes plus vifs remerciements à Roger PELLETRET, Chef de la Division Domotique et Physique Énergétique, dont les conseils et les encouragements ont rendu possible l'aboutissement de ce travail. Ces années de travail en commun ont été agrémentées par ses qualités d'animateur et ses compétences. Qu'il trouve ici l'expression de mon amitié sincère.

Je remercie Patrice POYET, Chef de la Division Bases de Connaissances, pour avoir rendu possible l'implémentation efficace du modèle de l'ESI en jetant les bases de la plate-forme logicielle puissante et novatrice utilisée.

Merci à Eric BRISSON pour son apport concernant les méthodes de spécification formelle ainsi que pour sa disponibilité.

Je suis reconnaissant à Monsieur Jean BRAU, Professeur à l'INSA de Lyon, qui me fait l'honneur de présider ce jury ainsi qu'à Messieurs Alain CORDIER, Professeur à l'Université de Toulouse, et Jean-Claude MANGIN, Professeur à l'Université de Savoie, d'avoir accepté la lourde tâche de rapporteurs.

Je remercie Messieurs Claude-Alain ROULET, chercheur au LESO, et Roland FAUCONNIER , chercheur à la FNB, qui ont eu la gentillesse d'accepter de participer à ce jury.

Merci à Nicole BENSOUSSAN, Nicole KEISER et Thierry MOTTE pour leur aide efficace dans la préparation des modalités de la soutenance et la réalisation des copies de ce document.

RESUME

La simulation numérique est un outil d'aide à la conception et à l'audit encore peu utilisé dans le secteur du bâtiment par comparaison avec d'autres secteurs d'activité (électronique, espace, automobile...).

Parallèlement, dans le milieu de la recherche connexe au bâtiment, des codes de simulation permettant de répondre à un ensemble de problèmes scientifiques et techniques ont été développés. Malgré leurs performances sur le plan numérique, ces codes manquent d'un certain nombre de fonctionnalités afin que leur utilisation soit simple et productive (assistance à l'utilisateur, vérification de la cohérence, échange de données...).

Dans ce but, ce travail de recherche s'attache à la conception d'un Environnement de Simulation Intelligent (ESI) alliant les performances numériques des codes de simulation existants aux possibilités offertes par les systèmes à base de connaissances. Il s'agit de développer au sein de l'ESI une partie symbolique et une partie numérique : la partie symbolique agit alors comme un serveur sémantique qui d'une part supporte des opérations de haut niveau d'abstraction (partage de données entre différents codes de simulation, modélisation du raisonnement...) et, d'autre part, pilote la partie qui assure la résolution numérique du système étudié.

Après une brève présentation du processus de modélisation/simulation/analyse des résultats et des modalités du dialogue Homme/Machine associées à l'utilisation des ordinateurs dans ce processus, les composants de l'ESI sont décrits suivant une méthode de spécification formelle basée sur une approche orientée objet.

Une première application au secteur du bâtiment, implantée au sein d'une plate-forme logicielle pour la génération de systèmes à base de connaissance développée au CSTB et utilisant le modèle de données intégré du bâtiment issu du projet européen COMBINE, est présentée.

Mots-clés : Modélisation - Simulation - Bâtiment - Bases de connaissances - Interface Homme/Machine - Conception Orientée Objet.

ABSTRACT

Numerical simulation is a powerful design and audit tool. But in the building industry, it is not as widely spread as in other industries (electronics, car industry, space industry...).

Meanwhile, in building related research centers, various simulation codes have been developed. These codes are generally powerful on the numerical level and can cope with a large range of building related issues.
Nevertheless, these codes lack some features in order to make their use simple and productive (user guidance, coherency check, share of data...).

The aim of this work is to design an Intelligent Simulation Environment (ISE) taking advantage of both numerical performances of existing simulation codes and symbolic performances of knowledge based systems.

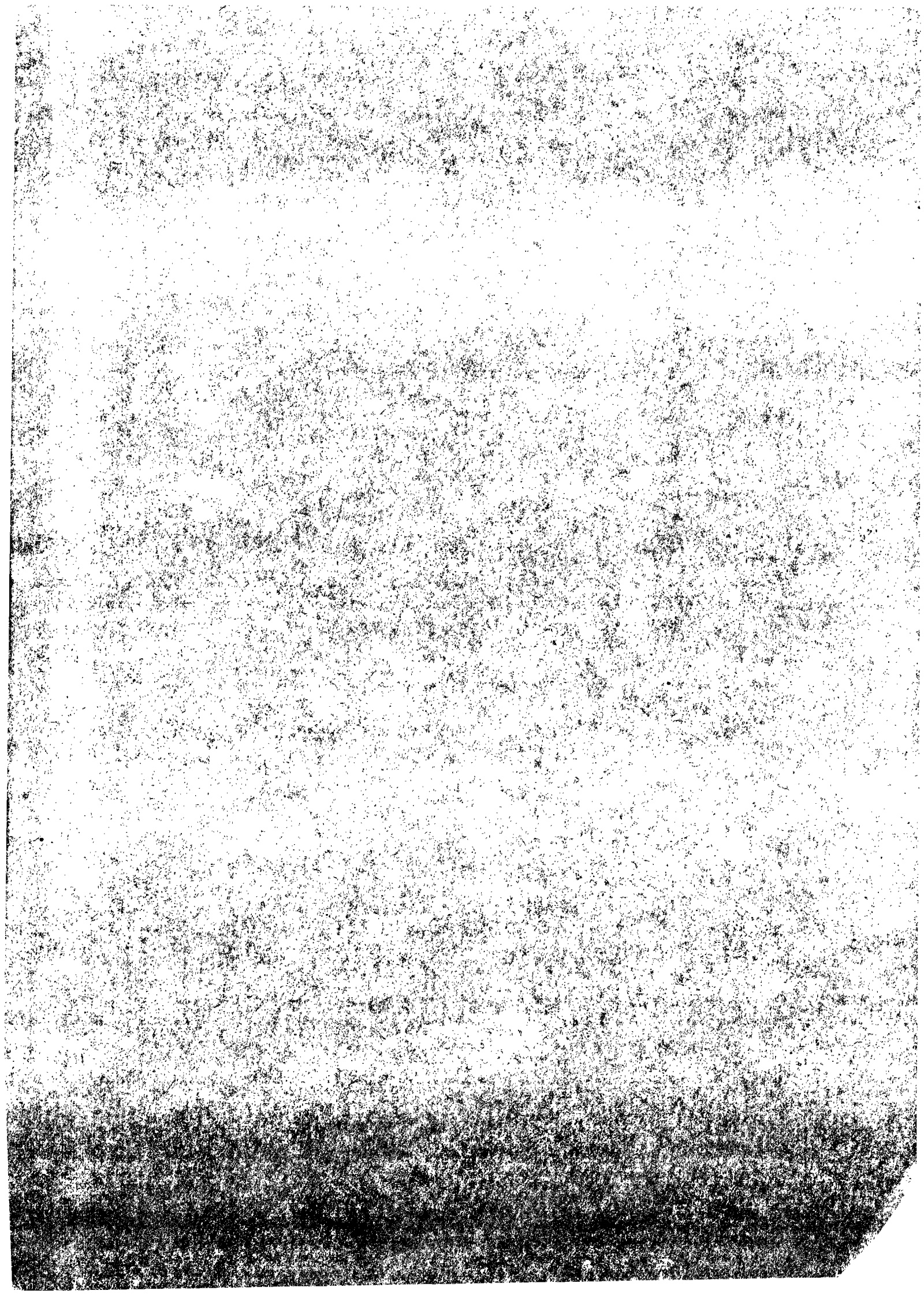
The symbolic part of the ISE will act as a semantic server taking in charge operations on a high level of abstraction (sharing of data, reasoning strategies...) and controling the numerical part of the ISE which will act as a number cruncher.

After a short presentation of the modelling/simulation/result analysis process and of the possible Human/Computer interactions, the components of the ISE are described using an object oriented formal specification method.

A first application to the building field is then presented. This application is implemented within a knowledge modelling software platform developed in the CSTB and uses the Integrated Data Model resulting from the IDM task of the CEC project COMBINE.

Key-words : Modelling - Simulation - Building - Knowledge Based Systems - Human/Computer dialogue. Object Oriented Design.

SOMMAIRE



INTRODUCTION

Le présent travail vise à spécifier les fonctionnalités d'un outil informatique permettant d'améliorer la productivité des utilisateurs de codes de simulation et la qualité des études réalisées avec ces codes. Il s'agit d'un outil (Environnement de Simulation Intelligent) qui permet d'intégrer des codes de simulation différents et de faciliter le partage des données entre ces codes. Ce partage des données permet, d'une part, d'alléger l'étape de saisie d'un projet de simulation et, d'autre part, d'assurer la cohérence et l'unicité des données d'un projet de simulation.

1. CONTEXTE

Le point de départ du présent travail est le constat suivant : il existe, dans le domaine du bâtiment, de nombreux codes de simulation. Ces codes peuvent traiter plusieurs problématiques liées au bâtiment (performances énergétiques, conditions de confort, transferts d'air...) et ceci à des échelles différentes (globale, détaillée...).

Ces codes, pour la plupart issus de la recherche, sont souvent performants. Néanmoins leur utilisation par les professionnels du bâtiment (bureaux d'étude, architectes, gestionnaires...) reste limitée. Ceci est principalement dû aux raisons suivantes :

° la description par les utilisateurs du problème à simuler doit être traduite en des termes et suivant une syntaxe compréhensibles par les codes de simulation (*langage d'entrée*). Chaque code de simulation possède son langage d'entrée propre qui est généralement plus proche des langages informatiques que du (des) langage(s) utilisé(s) par les professionnels du bâtiment. L'utilisation des codes de simulation nécessite donc un temps d'apprentissage du langage d'entrée et, éventuellement, un temps de ré-apprentissage pour les utilisateurs occasionnels. La traduction du problème physique traité en langage d'entrée est une étape généralement jugée par les professionnels du bâtiment comme redondante et difficile. Le même problème se pose pour les sorties du code de simulation (langage de sortie) où l'utilisateur est amené à interpréter les sorties pour en tirer des conclusions quant au problème physique traité ;

° l'échange de données entre les codes de simulation existants est difficile. Ceci est d'autant plus mal perçu par les utilisateurs que, d'une part, la saisie des données caractérisant le bâtiment peut être fastidieuse et, d'autre part, ces données sont relatives aux mêmes composants technologiques ;

° les utilisateurs des codes de simulation dans les professions du bâtiment ont des exigences différentes que ceux des chercheurs : ils s'attendent à utiliser des outils ergonomiques qui prennent en charge tout ou partie des tâches répétitives, et qui peuvent les assister durant les tâches de conception (par des aides pertinentes, des vérifications du travail effectué, des propositions de solutions...). Ces exigences ne sont pas prises en compte par les codes de simulation existants.

2. OBJECTIFS

Nous nous proposons, dans un premier temps, de définir les spécifications d'un Environnement de Simulation Intelligent (ESI) et ceci indépendamment du domaine traité. Il s'agit d'un outil qui peut être utilisé dans tous les domaines où la simulation est un moyen d'investigation du comportement de systèmes technologiques complexes soumis à certaines sollicitations. Cette investigation peut avoir des objectifs aussi variés que l'audit, l'aide à la conception (explorer différentes alternatives), l'analyse de sensibilité...

Nous nous proposons ensuite d'appliquer ces spécifications au secteur du bâtiment en y intégrant un modèle de données du bâtiment et de ses équipements. L'ESI ainsi adapté au secteur du bâtiment peut alors servir de base pour la réalisation d'outils "d'aide à la décision" par une prise en compte d'une démarche multi-contraintes associée à des codes de simulation variés permettant diverses évaluations techniques (thermique, acoustique, aéraulique...).

Les caractéristiques d'un ESI sont les suivantes :

° il permet d'établir un dialogue conceptuel avec ses utilisateurs : les langages d'entrée et de sortie utilisent les même concepts que ceux manipulés par les utilisateurs pour décrire leur problème physique. Ce dialogue conceptuel permet de réduire l'effort nécessaire à la traduction en langage d'entrée du problème à traiter et l'effort nécessaire à l'analyse du langage de sortie ;

° il est capable de reconnaître le contexte des actions de l'utilisateur et par conséquent de l'assister en lui proposant des aides pertinentes, des vérifications de la cohérence des actions... ;

° il se base sur une modélisation profonde des entités manipulées par les codes de simulation ce qui lui permet d'une part d'intégrer facilement des codes de simulation variés et, d'autre part, pour un projet de simulation donné, de faciliter l'échange des données communes à ces différents codes.

La réalisation d'un ESI dans le domaine du bâtiment sera sans doute un vecteur important pour augmenter la dissémination des codes de simulation dans les professions du bâtiment et, par conséquent, pour mieux valoriser les travaux issus de la recherche et améliorer la qualité des études menées par ces professions.

Bien que n'ayant pas l'exclusivité de ces préoccupations, qui se posent en termes semblables dans différentes équipes de recherche, l'apport de ce travail de thèse se situe au niveau de la méthodologie et de l'intégration des concepts développés dans le contexte plus général des recherches menées au CSTB, dans le cadre d'une part de l'élaboration d'un modèle de données intégré pour le bâtiment [DEL92] et, d'autre part, du développement d'une plate-forme logicielle pour la génération de systèmes à base de connaissance [PBD92].

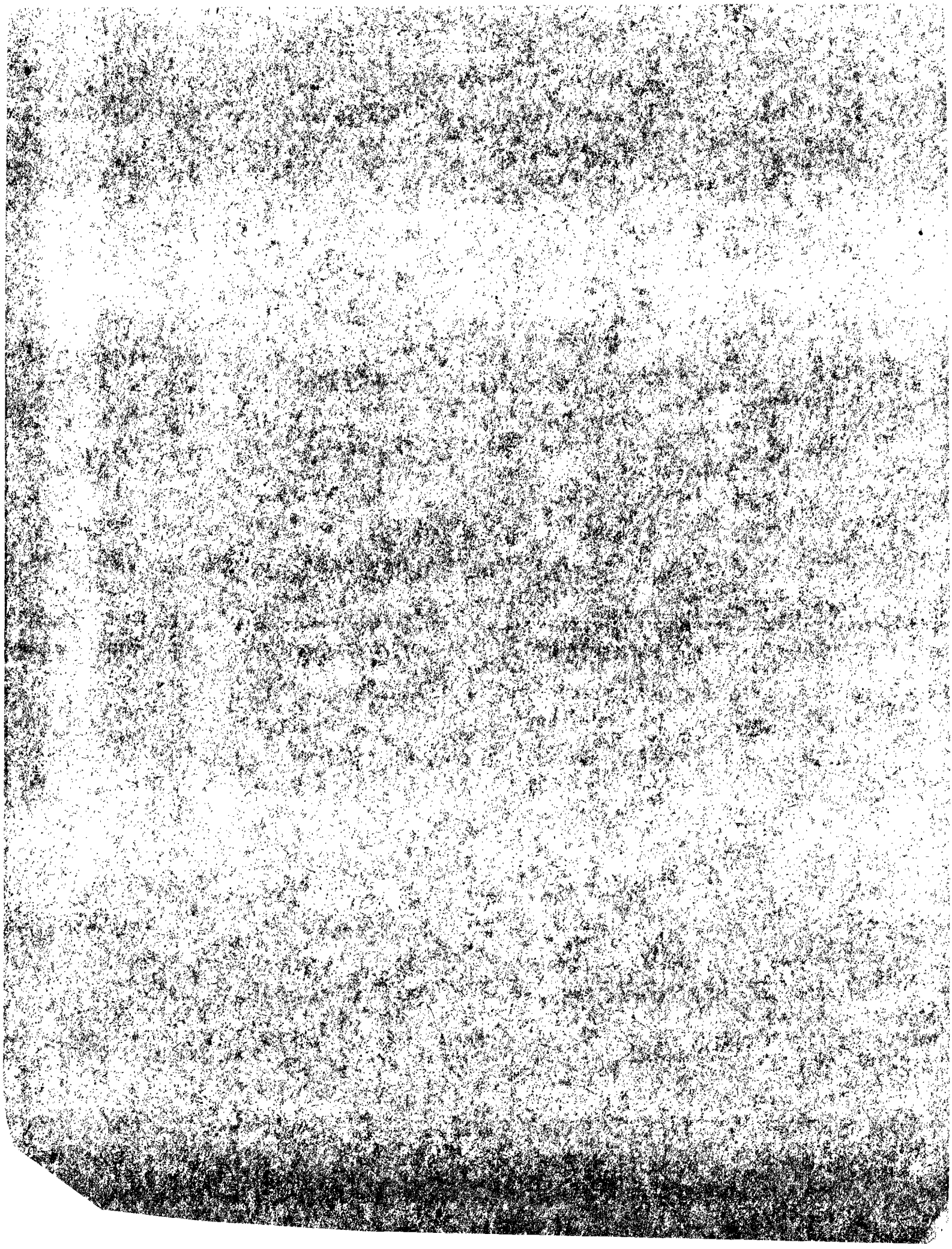
3. PRESENTATION DU DOCUMENT

Un travail sur le développement d'environnements intelligents pour la modélisation et la simulation du bâtiment est un travail pluridisciplinaire où interviennent :

- l'ergonomie et les sciences cognitives;
- le domaine applicatif, en l'occurrence la modélisation et la simulation du bâtiment ;
- la modélisation des connaissances et les techniques informatiques utilisées.

Ce document reprend cette pluridisciplinarité. Le premier chapitre fait le bilan des études relatives à la conception des interfaces H/M dans le domaine de l'ergonomie et des sciences cognitives. Le deuxième chapitre définit les différentes méthodes de travail dans le processus de modélisation et de simulation du bâtiment. Le troisième chapitre correspond à une spécification d'un ESI indépendamment du domaine. Le quatrième chapitre présente le développement d'un ESI adapté au secteur du bâtiment.

N.B. Afin de faciliter la lecture de ce document, un glossaire reprenant la définition des notions utilisées y est joint. Les mots référencés dans le glossaire apparaissent en *italique* dans le texte.



CHAPITRE I : DIALOGUE HOMME / MACHINE

INTRODUCTION

L'un des aspects qui souligne le mieux le besoin d'avoir des interfaces Homme/Machine pertinentes pour les codes de simulation est l'aspect cognitif du processus de modélisation / simulation (la façon avec laquelle l'utilisateur perçoit et raisonne sur un modèle) : dans ce processus, l'utilisateur n'agit ni sur le système réel, ni sur le modèle abstrait (mathématique) mis en œuvre dans le logiciel. Il agit sur un "objet" à un niveau d'abstraction intermédiaire entre le concret et l'abstrait et qui n'a pas d'équivalent en dehors de la simulation [Heb75] : l'utilisateur peut "agir" sur le modèle à l'aide du clavier et/ou de la souris ou tout autre moyen de communication afin de spécifier les paramètres de son problème, le modèle "réagit" par des informations affichées à l'écran.

Puisque toute action entraîne une réaction, l'image à travers l'interface du modèle mathématique devient une entité ayant son existence propre, et la réflexion de l'utilisateur va porter sur ce niveau intermédiaire d'abstraction.

La démarche cognitive de l'utilisateur dans un processus de modélisation/simulation est donc fortement liée aux informations fournies par l'interface Homme/Machine.

Dans le cadre de la spécification d'un ESI, une meilleure compréhension du dialogue Homme/Machine (H/M) est donc nécessaire (le dialogue H/M est défini comme l'échange d'information entre un outil informatique et ses utilisateurs via une interface H/M).

Deux méthodes sont possibles pour évaluer ce dialogue :

- la première est une approche empirique qui constitue un contrôle a posteriori des outils évalués. Elle repose sur l'observation du comportement des utilisateurs en situation réelle et l'interprétation de ce comportement afin de déterminer les origines et les causes des difficultés observées. Une fois ces causes définies, des solutions pour les réduire sont intégrées dans l'outil. Cette démarche itérative suppose qu'il existe une expérience de l'utilisation de l'outil évalué qui, par définition, n'existe pas dans un contexte de conception ;

- la deuxième approche est une évaluation a priori des caractéristiques ergonomiques des outils informatiques et nécessite une approche analytique. Elle repose sur une modélisation formelle du dialogue H/M permettant de prédire les performances des utilisateurs et le comportement de l'interface.

L'évaluation a priori semble la plus intéressante puisqu'elle permet une prise en compte de l'utilisateur dès la phase de conception de l'interface H/M ce qui évite des améliorations itératives et souvent coûteuse. Néanmoins, pour que cette évaluation soit pertinente, il est nécessaire d'avoir des modèles du dialogue H/M reproduisant simultanément le comportement de l'utilisateur, de l'interface et de leurs interactions.

Or, il n'existe pas à l'heure actuelle de modèle intégrant les différents éléments du dialogue H/M ; il existe des modèles reproduisant une partie de ses éléments : des modèles de l'utilisateur (cf. § 1), des modèles de l'activité qui décrivent les tâches réalisées par l'utilisateur (cf. § 2) des modèles du langage d'interaction H/M (cf. § 3) et des modèles de l'interface H/M (cf. § 4).

Par ailleurs, pour chacune de ces quatres classes de modèles, la modélisation peut s'effectuer des niveaux de description différents : les modèles peuvent décrire des tâches de bas niveaux (gestion de l'écran, utilisation du clavier...) ou décrire la démarche cognitive des utilisateurs.

Nous nous proposons dans ce chapitre d'étudier les modèles existants permettant une évaluation a priori des caractéristiques ergonomiques des outils informatiques afin d'en tirer des conclusions quant au choix du/des modèles adaptés au cas spécifique des outils de simulation.

1. MODELISATION DE L'UTILISATEUR

Les modèles de l'utilisateur permettent de reproduire et/ou de prédire le comportement moyen des utilisateurs d'un outil informatique. Ils sont basés sur une description de la boucle "perception-action" de l'utilisateur. Dans ce paragraphe, un bref historique de l'évolution de ces modèles est présenté et les caractéristiques des modèles les plus marquants sont soulignées.

1.1. SYSTEME DE TRAITEMENT DE L'INFORMATION

Une des premières formalisations du comportement de l'utilisateur face à la machine est le modèle du Système de Traitement de l'Information (STI). Le modèle STI a été développé par Newell et Simon à partir des années 1960 [BHT85]. La structure d'un STI est celle d'une machine de Turing universelle : elle comprend un processeur, des mémoires (de long et de court terme), et, pour communiquer avec le monde extérieur, des récepteurs et des effecteurs (organes à l'origine des réactions). L'instanciation de ce modèle dans le cas d'un "système humain" ou d'un "système informatique" est immédiate. Ce modèle est à la base du modèle du Processeur Humain proposé par Card [CMN83].

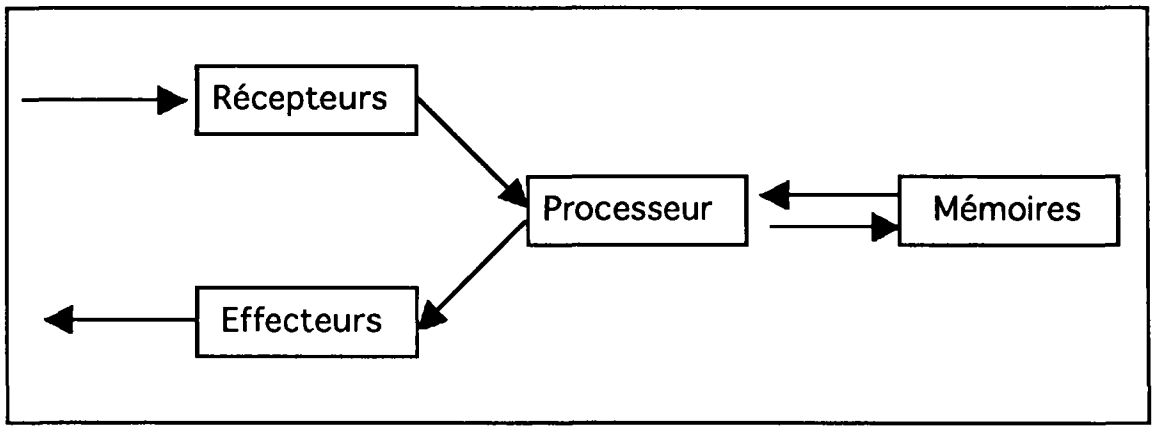


Figure I.1 : Système de Traitement de l'Information.

1.2. MODELE DU PROCESSEUR HUMAIN

Le modèle du Processeur Humain (basé sur le modèle STI) est composé de trois processeurs : le processeur perceptuel (éventuellement découpé en processeur visuel et processeur auditif), le processeur cognitif et le processeur moteur. A chacun de ces trois processeurs est rattachée une mémoire. La mémoire et les performances du processeur sont caractérisées par des paramètres.

Pour les mémoires, ces paramètres sont :

° m, la capacité de la mémoire en éléments d'informations ;

° d, la dégradation d'un élément définie comme étant le temps au bout duquel la probabilité de retrouver cet élément est inférieure à 0,5 ;

° k, le type d'information mémorisée.

Pour un processeur, le paramètre essentiel est :

° t, le cycle de base du processeur qui inclut le cycle d'accès à la mémoire associée.

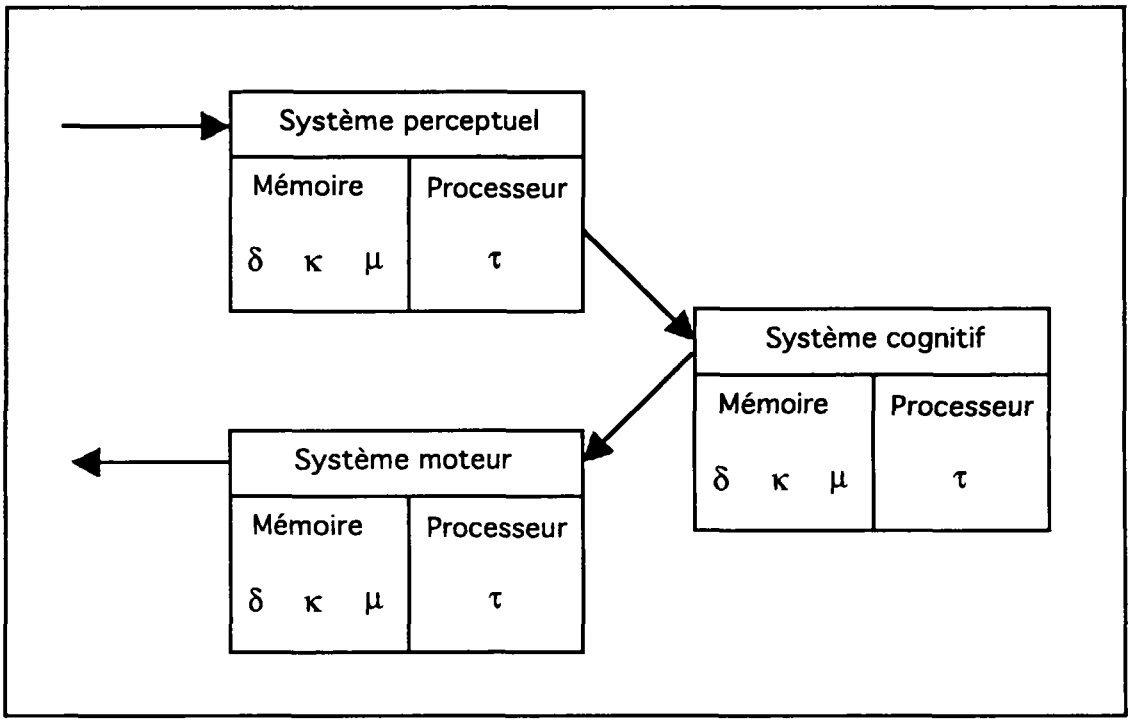


Figure I.2 : Modèle du Processeur Humain.

Les principes sous-jacents au fonctionnement de ce modèle tels qu'ils ont été énoncés par Card (cité dans [Cou87b]) sont listés ci-après.

° Principe de codage : des opérations spécifiques de codage appliquées sur le "perçu" déterminent ce qui sera sauvegardé en mémoire ; et ce qui sera sauvegardé détermine les clés d'accès qui seront utilisées.

° Principe de discrimination : la difficulté de restitution est déterminée par les candidats potentiels en mémoire relatifs aux clés d'accès.

° Principe de rationalité : un utilisateur agit pour atteindre son but suivant une démarche rationnelle, prenant en compte la structure de la tâche et les informations disponibles, et limitée par l'étendue de ses connaissances et de ses capacités de raisonnement.

° Principe de l'espace de problèmes : l'activité rationnelle engagée par les utilisateurs pour résoudre un problème peut être décrite en termes : (1) d'ensemble d'états de la connaissance, (2) d'opérateurs pour passer d'un état à un autre, (3) de contraintes sur l'application de ces opérateurs et (4) de règles pour décider l'enchaînement des opérateurs.

Suivant le modèle STI et le modèle du Processeur Humain, le comportement de l'utilisateur face à la machine peut être divisé en trois tâches : la perception, la cognition et l'action (le but du concepteur d'interface est dans ce cas de minimiser l'effort nécessaire pour l'accomplissement de chacune de ces tâches réalisées séparément ou en parallèle). Les concepts qui servent à l'étude des aspects de perception et d'action découlent surtout des recherches en ergonomie tandis que ceux qui servent pour l'étude des aspects de cognition découlent des recherches en psychologie cognitive.

1.2.1. Perception

La perception désigne le processus où un stimulus (dans ce cas provenant de la machine) est reçu par un système sensoriel, classifié pour être traité dans le stade de cognition. Les stimuli dans le dialogue H/M sont principalement visuels mais les stimuli auditifs sont aussi présents (bruit des touches du clavier, accès disque, sorties vocales...) ainsi que les stimuli tactiles (souris, clavier, écran tactile...). Le cycle de base du processeur en perception visuelle est de l'ordre de 100 msec (ce qui représente le temps entre l'envoi du stimulus et l'instant où l'individu a la sensation de percevoir). Ce cycle est fonction de plusieurs paramètres (couleur, clignotement, taille, mouvement ...). Le concepteur d'interfaces doit conjuguer ces paramètres afin d'attirer l'attention de l'utilisateur sur les parties de l'écran qu'il juge importantes pour la tâche réalisée.

1.2.2. Action

La phase d'action se déroule quand l'utilisateur ayant reçu, reconnu et décidé de réagir à des stimuli, formule la réponse en termes d'actions. Dans le cadre de l'interaction H/M, ces actions concernent essentiellement des mouvements qui manipulent des

unités d'échange (claviers, dispositifs de désignation...). La loi empirique de Fitt* montre que le temps T nécessaire pour positionner la main (ou le curseur de la souris) sur une cible dépend de la précision requise, c'est à dire du rapport entre la distance et la dimension de la cible.
En fait, les actions de l'utilisateur dépendent aussi du retour d'informations ("feed-back") qui se fait de façon continue. Par exemple, l'affichage de curseur sur l'écran est une source informative permettant aux actions de l'utilisateur de converger plus rapidement vers la cible.

1.2.3. Cognition

La psychologie cognitive traite de la manière avec laquelle l'information est acquise, organisée et recherchée (cognition). Les utilisateurs de systèmes informatiques passent par la phase de cognition non seulement lors de l'apprentissage de l'utilisation d'un nouveau système mais aussi durant son utilisation courante. Augmenter la productivité de ces utilisateurs passe donc par une meilleure compréhension des divers composants de la phase de cognition.

La mémoire du système cognitive est composée d'une mémoire à court terme (ou mémoire de travail) et d'une mémoire à long terme.

La mémoire à court terme contient des résultats intermédiaires du raisonnement, des informations en provenance du système perceptuel et un sous-ensemble de connaissances extraites de la mémoire à long terme.

La mémoire à long terme stocke les connaissances pour une utilisation future sous forme d'unités cognitives ("chunks"). La nature de l'unité cognitive de base dépend de l'individu (par exemple, "IA" est une unité cognitive de base de type "acronyme" pour les personnes qui en connaissent la signification). Les unités cognitives peuvent être regroupées dans des entités plus grandes et peuvent être liées à d'autres unités (par exemple, l'unité "pneu" est lié à l'unité "voiture" par une relation du type "est une partie de..."). Les réseaux sémantiques ont souvent été utilisés pour représenter la connaissance sous forme d'unités cognitives et de relations entre ces unités (pour plus de détails, se référer à [BF81]).

La capacité m de la mémoire à court terme est réduite. Les nouvelles informations stockées en mémoire de travail ont tendance à effacer les informations déjà présentes. Une estimation de cette capacité est donnée par Miller (cité dans [Cou87a]). Elle est de l'ordre de 7 ± 2 unités cognitives (suivant les individus et suivant la difficulté du problème). La dégradation des informations en mémoire de travail est rapide. Un élément qui n'est pas réactivé aura tendance à disparaître de cette mémoire.

La capacité m de la mémoire à long terme est infinie. La dégradation d d'un élément de cette mémoire est aussi infinie. Les échecs de requête d'informations en mémoire à long terme proviennent donc soit de l'absence d'une clé d'accès à l'information soit d'une utilisation d'une seule clé d'accès pour plusieurs unités cognitives. La présence en

---

* $T = I \log (D/S + 0.5)$ où I est une constante pour un utilisateur donné
D est la distance entre la position initiale et le centre de la cible S est la surface de la cible.

mémoire de ces unités cognitives différentes interfèrent alors avec la requête et empêchent son succès (on parle alors de "bruit").

Les incidences de ces constats sur la conception des interfaces H/M sont les suivants :

° il faut éviter de surcharger la mémoire de travail de l'utilisateur. Certains composants de l'interface (menus, valeurs par défaut ...) peuvent être utilisés comme des extensions de la mémoire de travail ;

° l'acquisition de nouvelles connaissances est d'autant plus facile qu'elles correspondent à des classes de connaissances connues (stockées en mémoire à long terme). Le corollaire de ce constat est le suivant : le concepteur d'interfaces doit s'attacher à utiliser des concepts proches de ceux déjà connus par les utilisateurs (l'utilisation de la métaphore du bureau pour l'interface du Macintosh en est un exemple) ;

° la charge de cognition dépend des connaissances de l'utilisateur. Ainsi, plus cette charge est réduite, plus l'utilisateur est un expert du domaine. A ce niveau, on fera la distinction entre les connaissances de l'utilisateur concernant le domaine applicatif (dans ce cas, on parlera d'expert et de novice) et les connaissances de l'utilisateur concernant l'outil informatique (dans ce cas, on parlera d'utilisateur averti et d'utilisateur non-averti) : les experts du domaine applicatif tendent à avoir une charge globale de cognition minimale ; d'un autre côté, les utilisateurs avertis tendent à minimiser la charge cognitive relative aux actions de bas niveau (recherche du nom de l'action à exécuter, déplacement de la souris...) et améliorent leurs performances au niveau de la perception et de l'action.

1.2.4. Evaluation du modèle STI et du modèle du Processeur Humain

L'avantage de ces modèles réside dans la formalisation de certaines des théories de la psychologie cognitive (introduction de paramètres quantifiables comme la capacité et la dégradation de la mémoire, loi de Fitt...). Ceci donne au concepteur d'interfaces le moyen de prédire certains des comportements des utilisateurs. Néanmoins, ces prévisions se limitent aux aspects de perception et d'action, le processus de cognition étant mal défini.

2. MODELISATION DE L'ACTIVITE

Les modèles de l'activité décrivent l'activité humaine en termes d'objectifs à atteindre, d'actions à réaliser pour atteindre l'objectif et d'évaluation des résultats des actions au regard de l'objectif initial. Les travaux les plus marquants sur la modélisation de l'activité sont la Théorie de l'Action développée par Norman [ND86] (cf. § 2.1) et le modèle GOMS élaboré par Card [CMN83] (cf. § 2.2).

2.1. THEORIE DE L'ACTION

L'hypothèse de base de la Théorie de l'Action est que la complexité de tout dispositif est essentiellement un problème cognitif. Les connaissances mises en jeu par un utilisateur lorsqu'il est confronté à un problème comportent deux représentations mentales :

° un modèle mental de la tâche à réaliser ;
° un modèle mental du dispositif utilisé.

Se basant sur ces modèles mentaux, l'utilisateur met en œuvre la réalisation de la tâche suivant un processus qu'on détaillera par la suite. La qualité de l'interface H/M dépend alors de sa compatibilité avec les représentations mentales élaborées par ses utilisateurs.

2.1.1. Qu'est ce qu'un modèle mental ?

Un modèle mental peut être défini comme une représentation conceptuelle d'un système réel. On part de l'hypothèse que le fondement psychologique de la compréhension est l'existence dans la pensée d'un *modèle opérationnel.* Aussi nous reprenons la définition de Johnson-Laird [JL83] concernant les modèles mentaux : "*si vous savez ce qui cause un phénomène, comment il se déroule, ce qui en résulte, comment on peut l'influencer, le contrôler, l'initialiser ou l'empêcher, alors d'une certaine manière vous le comprenez*".

Les modèles mentaux possèdent les caractéristiques suivantes :

° pour un même phénomène physique, plusieurs modèles mentaux peuvent être construits. Suivant l'acquis des utilisateurs, ces modèles possèdent des niveaux d'abstraction différents et manipulent des concepts différents (cf. Fig.I.3) ;

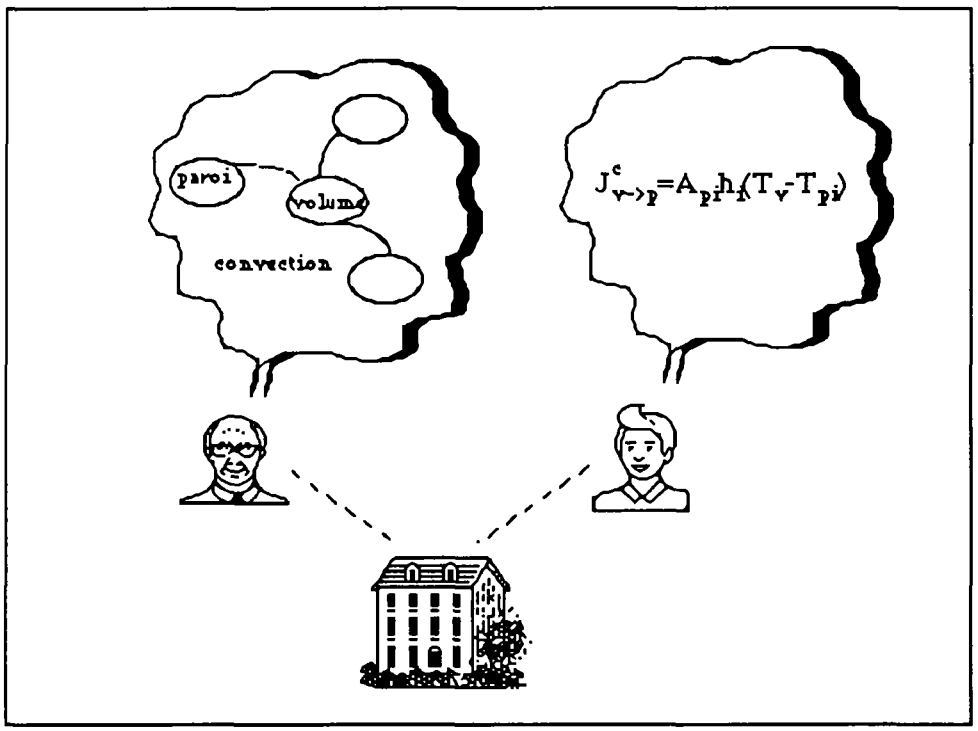


Figure I.3. Conceptualisations différentes d'un même objet physique.

° les modèles mentaux peuvent être plus ou moins proches de la réalité. Puisqu'aucun modèle (ni mental, ni mathématique) n'est capable de prendre en compte toute la complexité de la réalité [Heb75], il existe forcément un décalage entre l'objet réel et le modèle mental (par exemple : on peut construire un modèle mental de la télévision sans nécessairement comprendre l'électrodynamique quantique) ;

° les modèles mentaux ne sont pas figés dans le temps et passent par des phases de reconceptualisation.

Les modèles mentaux varient donc suivant les utilisateurs, et, pour un même utilisateur, varient dans le temps. Ils sont forcément une simplification plus ou moins prononcée d'une réalité complexe.

Suivant la théorie de l'Action, l'utilisateur construit un modèle mental de l'outil qu'il manipule. Ce modèle peut correspondre au fonctionnement de l'outil comme il peut être erroné. Des expériences sur différents groupes d'utilisateurs prouvent que l'existence d'un modèle mental fonctionnel de l'outil entraîne une utilisation plus aisée.

Une de ces expériences menée à l'université de Michigan (détaillée dans [KB84]), a comparé les performances de deux groupes d'utilisateurs d'un panneau de contrôle d'une machine outil. Le premier groupe avait reçu une description du fonctionnement interne du panneau avant d'entamer une formation sur les procédures de manipulation externe (en boîte noire). Le deuxième groupe a uniquement reçu la formation sur la manipulation externe. Les résultats de cette expérience montrent que le premier groupe :

° a appris les procédures de manipulation plus rapidement (28% d'écart entre le temps moyen d'apprentissage du premier groupe et celui du deuxième groupe) ;

° a exécuté ces procédures plus rapidement (+17%) ;

° a gardé en mémoire le déroulement des procédures plus longtemps (+19%) ;

° a optimisé les procédures plus souvent (+400%). Ce chiffre est le plus surprenant : en fait il reflète la différence entre les priorités de l'expert d'un domaine et celles d'un novice. Pour le novice la priorité se situe au niveau de l'opérationnalité, tandis que, pour l'expert, l'optimisation des procédures prime sur leur opérationnalité.

Cette expérience montre l'intérêt d'avoir un modèle mental fonctionnel (non-erroné) des outils manipulés. La prise en compte de cette dimension, dès la phase de conception des interfaces H/M (en évitant les exceptions, en utilisant des métaphores...), rend la construction de ces modèles par les utilisateurs plus aisée.

Un autre travail expérimental réalisé par Kellog et Breen [KB83] sur un système de formattage de texte (gestion des paragraphes, indentation...) montre que le modèle mental des utilisateurs non-avertis est basé sur des ressemblances de surface (présentation à l'écran, relations sémantiques entre les commandes...) ce qui organise la majorité des concepts d'une façon aberrante. Par contre, les utilisateurs avertis classent les concepts selon des critères fonctionnels et établissent des groupements cohérents.

Le corollaire qui découle de se constat est le suivant : pour améliorer la qualité des interfaces H/M, les ressemblances de surfaces ne doivent pas être fortuites mais doivent refléter des ressemblances fonctionnelles de façon à éviter les confusions potentielles.

2.1.2. Les aspects d'une tâche

Le processus cognitif qui conduit à la réalisation d'une tâche comprend les sept étapes suivantes (cf. Fig.I.4) :

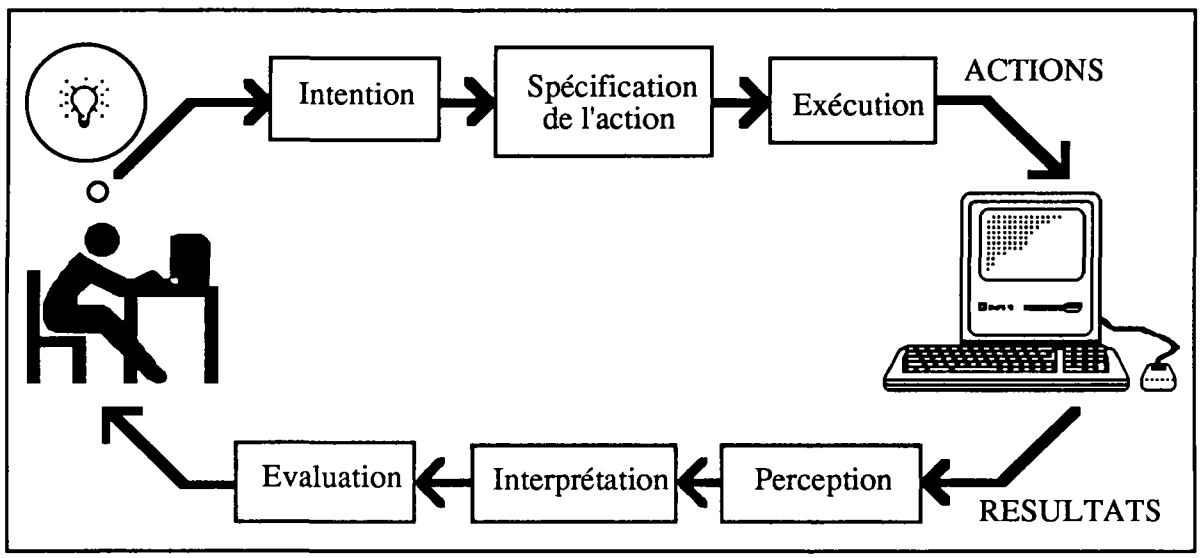


Figure I.4. Réalisation d'une tâche suivant la Théorie de l'Action.

° l'établissement d'un but : un but est une représentation mentale (il peut y avoir une différence entre l'état du système et la représentation que se fait l'utilisateur de cet état) de la distance qui sépare un état final du système d'un état initial.
La représentation de l'état final est élaborée en fonction des préférences et des méthodes de travail de l'utilisateur.
L'état initial du système est une représentation mentale construite par l'utilisateur à partir des variables "physiques" (exemple : forme du curseur, état des icônes, état de l'écran...).
La comparaison entre cet état initial (présent) et l'état final (désiré) amène l'utilisateur à établir un but (sous l'application MacDraw, un exemple de but est "rapprocher le cercle du carré") ;

° expression d'une intention : le but est traduit en intention exprimée dans le langage d'interface (exemple : "déplacer le cercle") ;

° définition d'une séquence d'actions : l'enchaînement des actions entraîne la satisfaction de l'intention (exemple : "sélectionner le cercle" et "déplacer le cercle") ;

° exécution de l'action : l'exécution des actions est réalisée par manipulation de dispositifs d'entrée (clavier, souris...) ;

° perception des effets de l'action : l'exécution des actions entraîne une modification, perçue par l'utilisateur, de l'état du système. Cette perception utilise des variables physiques (exemple : les coordonnées du cercle n'ont pas été modifiées) ;

° interprétation : les variables physiques perçues sont interprétées afin d'en dégager une représentation mentale de l'état du système (exemple : les coordonnées du cercle n'ont pas été modifiées car le cercle est groupé avec le carré) ;

° évaluation : l'état final du système est évalué en fonction du but. Cette évaluation peut entraîner une nouvelle itération.

Pour représenter la différence de niveau d'abstraction entre les concepts manipulés par l'utilisateur et les langages d'entrée et de sortie de l'interface, Norman introduit l'image de gouffre [ND86]. Un gouffre est la différence entre la représentation mentale du problème et la représentation externe imposée par le langage d'interface. L'objectif du concepteur d'interface est d'en réduire l'étendue.

Le gouffre de l'exécution exprime l'effort cognitif fournit par l'utilisateur pour passer de l'objectif à atteindre à l'action à accomplir sur les dispositifs physiques du système.

Le gouffre de l'évaluation traduit l'effort cognitif inverse pour passer de la perception d'un nouveau état du système à son interprétation en termes d'avancement vers l'objectif (cf. Fig.I.5).

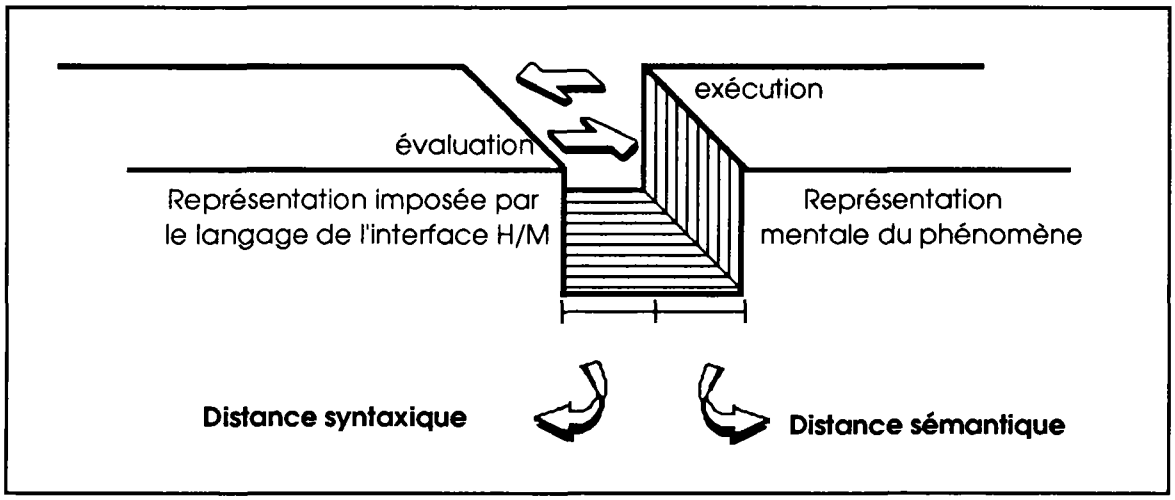


Figure I.5. Gouffre sémantique et syntaxique.

Ce gouffre, à l'exécution comme à l'évaluation, comporte en fait deux parties : une distance sémantique et une distance de forme.

La distance sémantique est celle qui sépare l'objectif visé par l'utilisateur de la signification des commandes nécessaires pour réaliser cet objectif (exemple tiré de l'éditeur de texte Word sur Macintosh : la distance entre l'objectif "supprimer un mot" et la signification de la commande "couper").

La distance de forme est celle qui sépare la signification des commandes dans le langage d'interface des actions à accomplir sur les dispositifs physiques du système pour exprimer ces commandes (exemple : la distance entre la signification de la commande "couper" et les actions "sélectionner le mot" et "taper ctrl X").

Une tentative de formalisation du concept de gouffre à été introduite par Streitz [Str87] de la façon de suivante :

- soit L(f) la réalisation par l'outil informatique d'une fonctionnalité f (par exemple "copier", "déplacer", "enregistrer"...) ;
- soit U(f) le modèle mental propre à l'utilisateur concernant la fonctionnalité f ;
- soit U(L(f)) la représentation cognitive qu'a l'utilisateur de la réalisation de la fonctionnalité f par l'outil.

Plus un système est cohérent, moins il y aura désaccord entre ces différents modèles de connaissances ; autrement dit, moins il y aura divergence entre L(f), U(f) et U(L(f)).

Le concepteur d'interfaces doit réduire les distances de forme et sémantique en proposant un dialogue qui correspond aux modèles mentaux manipulés. Ceci revient à trouver des représentations dont la signification et la forme se rapprochent le plus des concepts mis en œuvre par l'utilisateur.

2.1.3. Evaluation de la Théorie de l'Action

Cette théorie propose une description du processus cognitif de l'utilisateur. Néanmoins, cette description reste qualitative et ne peut pas servir à prévoir et à quantifier le comportement de l'utilisateur. D'autre part, la planification de l'activité de l'utilisateur et l'enchaînement des buts qu'il se fixe n'y sont traités que partiellement.

2.2. LE MODELE GOMS (GOALS, OPERATORS, METHODS, SELECTION RULES)

Le modèle GOMS, développé sur la base du paradigme STI, ne se contente pas de décrire la structure des tâches mais tente de modéliser la façon dont l'utilisateur s'y prend pour réaliser ses tâches.
Ce modèle est basé sur le Principe de rationalité cité ci-avant (cf. § 1.2) et repose sur une hypothèse simplificatrice qui suppose une utilisation sans erreurs de l'outil. Une application du modèle GOMS aux actions physiques réalisées sur le clavier a donné naissance au modèle KLM ("Keystroke Level Model") qui prédit le temps d'exécution d'une tâche routinière sur le clavier par un utilisateur averti ne commettant pas d'erreur.

2.2.1. Description du modèle GOMS

Les entités utilisés par le modèle GOMS sont les suivantes :

- les buts qui définissent l'état désiré. Les buts sont décomposés hiérarchiquement en sous-buts. Les feuilles terminales de la décomposition sont des opérateurs ;

- les opérateurs sont les actions élémentaires mises en jeu pour satisfaire le but poursuivi (actions de perception, motrices ou de cognition). Elles peuvent être caractérisées par des temps d'exécution ;

- les méthodes sont des procédures déjà acquises par l'utilisateur. Elle ne représentent pas un travail de planification en cours d'exécution. La limite entre les méthodes et les buts dépend de l'expérience de l'utilisateur concernant l'outil. Par exemple, dans un contexte d'édition de texte, le concept de copier/coller peut être une méthode pour un utilisateur averti (sa mise en œuvre ne nécessite plus une planification préalable). Par contre, ce même concept peut être considéré comme un but pour un utilisateur non-averti ;

- les règles de sélection qui déterminent le choix entre différentes combinaisons d'opérateurs aboutissant au même but final.

2.2.2. Evaluation du modèle GOMS

L'hypothèse simplificatrice de performances sans erreurs limite l'intérêt du modèle GOMS, la modélisation des erreurs étant une composante non-négligeable du modèle global (par exemple, pour une tâche d'édition de texte, 5 à 30% du temps total d'édition peut être attribué aux erreurs et à la durée de leur récupération [EN81]). De plus, le modèle GOMS ne prévoit pas la possibilité d'une planification de tâches parallèles.

3. MODELISATION DU LANGAGE D'INTERACTION H/M

Dans les modèles de l'activité, l'accent est mis sur la structure des tâches réalisées par l'utilisateur indépendamment du dispositif qui supporte l'interaction. Par contre, les modèles du langage d'interaction H/M rendent explicite la structure des contraintes induites par les dispositifs utilisés : l'interaction est modélisée comme un langage en couche grammaticales faisant passer du monde conceptuel et sémantique au monde perceptuel pour se traduire en actions sur les dispositif d'entrée et de sortie.

Les deux principaux modèles développés suivant cette approche sont le modèle "Action Language Grammar" (cf. § 3.1) développé par Reisner [Rei81] et "Command Language Grammar" (cf. § 3.2) développé par Moran [Mor81].

3.1. "ACTION LANGUAGE GRAMMAR" (ALG)

L'objectif de Reisner a été de construire un modèle qui rend possibles :

° la comparaison entre différentes alternatives de conception du point de vue de leur facilité d'utilisation ;

° l'identification des choix de conception qui peuvent conduire les utilisateurs à commettre des erreurs.

Les actions de l'utilisateur sont considérées comme un langage dont on peut rendre compte dans une grammaire formelle. Cette grammaire décompose hiérarchiquement le dialogue H/M d'une façon parallèle à la décomposition linguistique du langage naturel. Les différents niveaux de décomposition sont les suivants :

° le niveau conceptuel : c'est à ce niveau que sont définis les buts de l'utilisateur. Les questions à ce niveau sont : Quels types d'objets sont représentés dans l'ordinateur ? Quels sont leurs attributs ? Quels opérations peut-on effectuer sur ces objets (sans se préoccuper encore de la manière dont ces opérations peuvent être exécutées).

  Par exemple : avec un outil du style PaintBrush un but est "faire un dessin en couleur" ;

° le niveau sémantique : ce niveau représente les fonctionnalités de l'outil informatique. Il répond à la question : qu'est ce que chacune des opérations fait sur chacun des objets ? Le niveau sémantique décrit l'effet des actions de l'utilisateur sans indication sur la forme de ces actions ;

° le niveau syntaxique : à ce niveau les actions sont décrites sous forme de séquences d'unité d'information échangées, en entrée comme en sortie (par analogie au langage courant, ces séquences sont l'équivalent des phrases). Au niveau syntaxique, les séquences sont des symboles non-terminaux du langage d'interface (un symbole terminal du langage d'interface se traduit par une interaction élémentaire - clic de la souris, frappe de clavier... Un symbole non-terminal peut être décomposé en une suite de symboles terminaux).

  Par exemple : *<dessin en couleur> := <choisir une couleur> ET <choisir une forme>*
  OU *<choisir une forme>ET<choisir une couleur> ;*

° le niveau lexical : ce niveau est celui des unités d'information élémentaires de l'interaction (par analogie au langage courant, c'est le niveau des mots). Ces unités

d'informations sont les plus petites à la quelle le système informatique puisse attribuer une signification (mot, icône, nombre, coordonnées...).

Les séquences du niveau syntaxique sont décomposé au niveau lexical en une suite d'interactions élémentaires avec les dispositifs d'entrée/sortie (clavier, souris, écran...). Ces interactions élémentaires sont les symboles terminaux du langage d'interface (généralement représentées par des majuscules).

Contrairement aux autres niveaux, le niveau lexical dépend de l'environnement matériel et logiciel utilisé.

Par exemple : *<choisir une couleur>:= <CLIC CURSEUR DANS BLEU>*
OU <CLIC CURSEUR DANS ROUGE>...

L'intérêt du modèle ALG est qu'il permet une caractérisation quantitative du langage d'interface. Cette caractérisation utilise les trois métriques suivantes :

° le nombre global des symboles terminaux du niveau lexical reflète la complexité du langage (plus le nombre d'actions élémentaires possibles est élevé, plus le langage est complexe) ;

° la longueur des séquences du niveau syntaxique est un indice de la complexité des procédures (plus le nombre de commandes nécessaires pour réaliser une tâche donnée est élevé, plus les procédures sont complexes) ;

° le nombre de séquences redondantes fournit un indice sur le cohérence du langage d'interface (plus le nombre de procédures possibles pour réaliser une même tâche est élevé, moins le langage est cohérent).

Cette approche permet de formaliser certaines connaissances empiriques sur la conception des interfaces H/M. Un des exemples est la relation inversement proportionnelle entre le nombre de symboles terminaux et la longueur des séquences.

En effet, quand la taille du lexique est grande (par exemple, un nombre important d'outils dans une boîte à outils à la MacPaint), le concepteur d'interfaces a la possibilité de spécifier des procédures courtes pour la réalisation des tâches. Par contre, une taille du lexique moins importante (nombre plus réduit d'outils dont des outils complexes multi-fonctionnels) nécessite des procédures plus longues afin de permettre à l'interface de différencier les différentes actions (par exemple, distinction entre clic simple et double-clic pour différencier deux actions différentes du même outil).

Le modèle ALG présente néanmoins des limites. En effet, s'il permet des études entre plusieurs versions d'une interface H/M, il est insuffisant comme outil de spécification. Par exemple les critères de complexité et de simplicité sont trop réducteurs : des tâches faciles à décrire peuvent être difficiles à réaliser. D'autre part, la dynamique du dialogue peut difficilement être représentée par ce modèle (récupération d'erreurs, annulation, tâches parallèles) ce qui en fait un outil non-adapté à la gestion des interfaces graphiques.

3.2. "COMMAND LANGUAGE GRAMMAR" (CLG)

Le modèle CLG est basé sur une structuration linguistique du dialogue, distinguant des caractéristiques de différents niveaux d'abstraction : Conceptuel (qui correspond aux niveaux des tâches et sémantique du modèle ALG), Communication (niveaux syntaxique et lexical) et Physique où sont spécifié les aspects matériels et les unités d'entrée/sortie. Le modèle propose des métriques d'évaluation de la cohérence de la conception susceptibles de prédire l'efficacité de l'interface par une estimation de la vitesse d'exécution (en utilisant des formules de type KLM) ou par une estimation du temps d'apprentissage au niveau de chaque composant.

En raison d'une certaine lourdeur dans la mise en œuvre (il ne permet pas d'évaluer un produit fini développé avec une autre méthodologie que "CLG") et de limitations inhérentes au modèle (impossibilité de spécifier le parallélisme entre les activités, pas de modélisation des constituants graphiques de l'interface), ce modèle a été peu utilisé en pratique. il a eu, en revanche, une influence importante sur les recherches dans le domaine.

4. MODELISATION DE L'INTERFACE

Les modèles de l'activité et les modèles du langage d'interaction H/M s'attachent respectivement à décrire l'enchaînement des actions de l'utilisateur pour réaliser une tâche et l'influence du dispositif de dialogue sur l'interaction H/M. Ils ne fournissent aucune indication sur l'interface à réaliser. Les modèles de l'interface cherchent par contre à décrire la structure et le comportement de l'interface.

Deux approches sont pour les modèles de l'interface H/M :

° une description des exigences que doivent respecter les interfaces H/M pour répondre aux attentes des utilisateurs (cf. § 4.1) ;

° une description des composants et des comportement de l'interface. Cette approche est associée à des modèles centrés objets puisque ces modèles permettent un répartition du dialogue sur les composants de l'interface : les états possibles de l'interface ne sont pas décrit a priori, seuls sont décrit les états possibles de ses composants (cf. § 4.2).

4.1. MODELES DE QUALITE

Les modèles de qualité font l'hypothèse que la facilité d'usage d'une interface peut être estimée dès les premières étapes de conception suivant que cette interface respecte ou non certaines règles de base. Ces règles sont les suivantes :

° cohérence : si le but est le même dans des contextes différents, la suite d'actions doit être identique (par exemple, la procédure des "couper/coller" doit être la même que se soit sous un éditeur de texte ou sous un logiciel graphique) ;

° réversibilité : les actions de l'utilisateur doivent pouvoir être annulées. Ceci doit être aussi valable pour les actions destructrices (suppression d'un fichier...) même si ce n'est possible que pour une durée limitée ;

° réduction de la charge cognitive : l'interface doit décharger la mémoire à court terme de l'utilisateur en proposant une assistance dans les tâches les plus courantes (par exemple utilisation de valeurs par défaut, de menus...) ;

° flexibilité : les fonctions de l'interface doivent pouvoir s'adapter aux préférences et aux méthodes de travail de l'utilisateur (par exemple, un utilisateur peut choisir de garder ou de supprimer les messages d'avertissement du type : remplacer les éléments de même nom par ceux sélectionnés ?). A ce niveau on peut distinguer les interfaces adaptables des interfaces adaptatives. Une interface est adaptable lorsqu'elle est modifiable sur une intervention explicite de l'utilisateur. Une interface est dite adaptative lorsqu'elle peut s'adapter automatiquement et dynamiquement à l'évolution des connaissances et des préférences de l'utilisateur.

° structuration du dialogue : le dialogue doit être hiérarchisé en niveaux de complexité de sorte que l'utilisateur non-averti puisse, dès le début, réaliser les tâches les plus courantes. Il pourra, progressivement, découvrir la façon d'optimiser ces tâches ou découvrir des fonctions plus "riches" mais moins faciles à mettre en œuvre (sous le Finder du Mac, les raccourcis claviers sont un exemple d'optimisation des tâches, et la fonction ⌥ et clic dans le carré de fermeture permettant la fermeture de toutes les fenêtres ouvertes est un exemple de tâche "riche").

L'approche des modèles de qualité est intéressante car elle met à la disposition du concepteur d'interfaces des règles directrices directement applicables. Par contre elle ne permet pas d'évaluer l'impact du non respect de ces règles. Ce défaut est d'autant plus grave, que la conception d'interfaces passe souvent par des compromis pour respecter certaines contraintes. Dans ce cas, le concepteur n'est pas assisté pour choisir entre les différentes options possibles celle qui perturbe le moins le déroulement de la tâche. Par exemple, la relation entre la cohérence et la flexibilité du dialogue est souvent inversement proportionnelle (respecter rigoureusement la cohérence implique une certaine rigidité). Dans ce cas l'impact du non-respect d'une de ces règles doit être pondéré en fonction de la criticité et de la fréquence de la tâche réalisée.

4.2. MODELES CENTRES OBJET

L'origine des modèles Centrés Objets est liée au modèle de Seeheim. Ce modèle, qui tient son nom du lieu où s'est réuni en 1983 un groupe de spécialistes qui s'était fixé comme but de répertorier et d'organiser la connaissance dans le domaine des interfaces H/M, a introduit le principe pilote de la conception modulaire des composants des applications interactives : ce principe stipule qu'une séparation entre l'application (ensemble de fonctions propres au domaine) et l'interface (logiciel qui assure la communication d'information entre l'application et l'utilisateur) permet une modification aisée de l'application interactive. La modification de l'interface peut alors se faire sans remettre en cause les fonctions de l'application et réciproquement). L'interface est alors considérée comme un serveur syntaxique (dont les fonctions sont de gérer les images à l'écran et de lire les données d'entrée des dispositif physiques) et l'application comme un serveur sémantique (dont la fonction est de gérer le contexte des actions de l'utilisateur).

Le modèle de Seeheim indique les principes de la construction d'une application interactive sans donner d'indications sur les méthodes de mise en œuvre. Les modèles d'interface centrés objet reprennent d'une part les principes de base du modèle de Seeheim en les enrichissant par une séparation entre la syntaxe et la sémantique répartie sur tous composants de l'interface et, d'autre part, prévoient des méthodes de mise en œuvre. Le modèle MVC présenté ci-après (cf. § 4.2.1) a été élaboré au PARC

(Paolo Alto Research Center) comme support au langage Smalltalk [Gol84]. Il a été le premier modèle d'interface centré objet et demeure la référence pour cette classe de modèles.

4.2.1. Le modèle MVC (Model, Vue, Controller)

Les principes de base de ce modèle sont les suivants :

° séparation entre la syntaxe et la sémantique du dialogue H/M : l'architecture d'une application interactive suivant un modèle MVC se compose en trois entités de base :

- le Modèle : qui désigne la sémantique du dialogue (représentation internes des fonctions de l'application) ;
- la Vue : qui définit la syntaxe concrète de ce dialogue (présentation externe du système à l'utilisateur, gestion des entrées/sorties) ;
- le Contrôleur : qui joue un rôle médiateur entre la Vue et le Modèle. Les actions de l'utilisateur sont transmises par la Vue au Contrôleur et, selon l'état courant, celui-ci appelle les fonctions concernées du Modèle.

° la répartition du dialogue H/M : ce qui signifie qu'un ensemble d'objets implante le dialogue sous forme distribuée. En particulier chaque objet maintient un état courant. A la réception d'un message, chaque objet est capable de déterminer son comportement en fonction de son état courant (par exemple, un composant "bouton" dans l'état "grisé" va ignorer les événements souris). Contrairement à la programmation procédurale, il n'existe pas de composant qui centralise l'information concernant toutes les combinaisons possibles du système. L'état global du système est défini par l'état des objets qui le constituent.

Pour illustrer ces propos, prenons l'exemple d'une application interactive du type tableau de bord permettant de visualiser et d'agir sur certains paramètres d'un phénomène physique. Un code de simulation permet de calculer l'évolution dans le temps des différents paramètres.

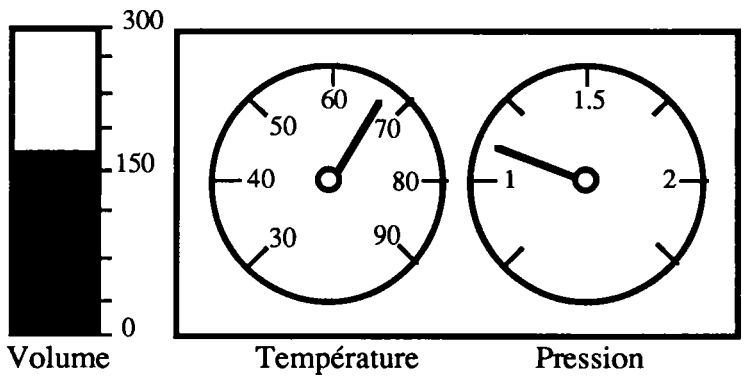


Figure I.6. Application interactive de type "tableau de bord" pour la compréhension de la loi des gaz parfaits.

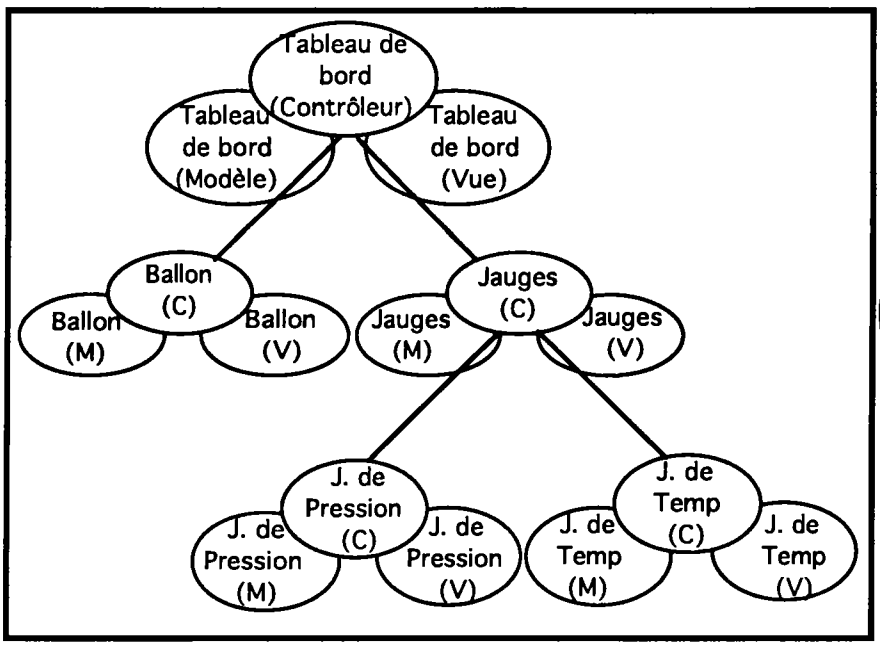
Si le but de l'interface est de pouvoir visualiser les valeurs des paramètres mais aussi de modifier ces valeurs interactivement à l'aide de la souris (Fig.I.6), les étapes de conception seront les suivantes :

° définir les objets qui entrent dans la composition du tableau de bord (jauges de température et de pression, indicateur de volume) ;

° définir une hiérarchie entre ces objets pour établir les relations d'héritage (Fig.I.7) ;

° pour chacun des objets, implanter la fonction qui va permettre de calculer la valeur de la variable qui s'y rattache (Modèle ou aspect sémantique) mais aussi l'aspect sous lequel on visualise cette variable (Vue ou aspect syntaxique) ;

° enfin, pour chacun des objets, implanter la partie Contrôleur qui répercute les conséquences des actions de l'utilisateur d'une part sur la partie Modèle de l'objet (exemple : clic avec la souris pour modifier la valeur de la température) et d'autre part sur tous les objets qui peuvent être affecté par ces actions (exemple : le changement de température peut entraîner un changement de pression et/ou de volume).



ECOLE NATIONALE DES PONTS-ET-CHAUSSEES
CENTRE PÉDAGOGIQUE DE DOCUMENTATION ET DE COMMUNICATION

Figure I.7. Entités manipulées par le modèle MVC de l'application "tableau de bord".

4.2.2. Evaluation des Modèles Centrés Objet

L'application de l'approche par objets pour la conception des interfaces H/M représente la tendance actuelle en recherche. Les avantages de cette approche pour le concepteur d'interfaces H/M sont l'extensibilité et la maintenabilité.

° L'extensibilité : pouvoir ajouter des objets à l'interface (exemple : ajouter un indicateur de la température extérieure). L'extensibilité est due à l'absence d'un organe centraliseur de la gestion du dialogue qui sera, par définition, difficile à modifier ;

° La maintenabilité : pouvoir modifier facilement l'aspect graphique ou le comportement interne d'un objet (exemple : remplacer la jauge par un lecteur digital). La maintenabilité est due à la séparation entre la sémantique et la syntaxique. Cette séparation permet de faire des modifications sur l'un des aspects sans effets de bord sur l'autre.

Pour l'utilisateur, l'avantage majeur de ces modèles est l'absence d'accès séquentiel à l'information et/ou aux écrans. Cette facilité provient de la répartition du dialogue sur les différents composants de l'interface ce qui permet de traiter les manipulations arbitraires de l'utilisateur.

BILAN ET CONCLUSION DU CHAPITRE 1

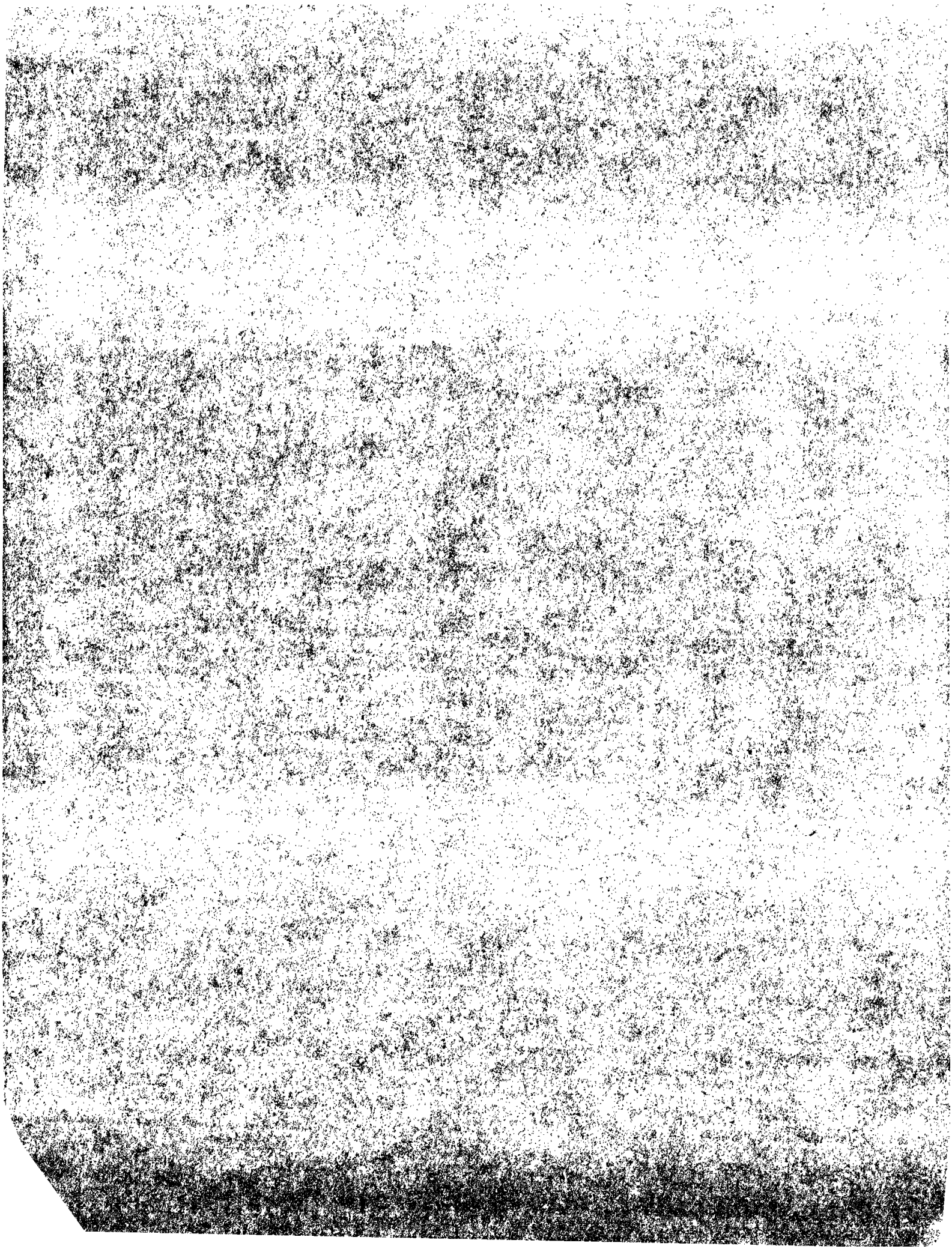
A l'issue de cette présentation, il apparaît que chacun des modèles étudiés possède des avantages mais ne modélise le dialogue H/M que d'une façon partielle. Par conséquent, la méthodologie suivie pour la conception du dialogue H/M des Environnements de Simulation Intelligents sera une méthodologie combinant les avantages de différents modèles présentés :

° elle est essentiellement basée sur le modèle MVC afin de tirer profit de la séparation entre la syntaxe et la sémantique ainsi que de la répartition du dialogue ;

° les règles des modèles de qualité servent de lignes directrices pour les choix de conception ;

° on retient du modèle ALG les métriques permettant de comparer différentes options de conception essentiellement pour les tâches de bas niveau ;

° pour les aspects sémantiques et conceptuels, la Théorie de l'Action fournit un cadre assurant une certaine conformité de l'interface aux représentations mentales de l'utilisateur.



CHAPITRE II : LE PROCESSUS DE MODELISATION / SIMULATION / ANALYSE DES RESULTATS

INTRODUCTION

La simulation est utilisée depuis longtemps en recherche et dans l'industrie, chaque fois que la construction d'un prototype d'étude s'avère trop coûteuse ou que l'étude mathématique usuelle s'avère trop longue ou trop complexe [Heb88].

Les premières simulations ont fait appel à des "modèles réduits" (ailes d'avions, coques de bateaux...), bientôt suivis par l'usage de calculateurs analogiques. Depuis une trentaine d'années, les simulations s'effectuent principalement sur ordinateur, même si certains laboratoires utilisent pour ce faire des calculateurs hybrides.

Les principaux buts de la simulation sont :

° prévoir avant la réalisation (aide à la conception, aide à la décision) ;

° vérifier / évaluer / comparer (contrôle, évaluation) ;

° optimiser / analyser la sensibilité (planification ...) ;

° expérimenter sans détruire, sans danger.

Dans ce chapitre, la notion de modèle utilisée par la suite est définie et, pour cette définition, le processus de modélisation / simulation / analyse des résultats est décomposé en tâches plus élémentaires. Cette décomposition du processus est nécessaire afin de spécifier les fonctions d'assistance que peut offrir l'ESI à ses utilisateurs durant chacune des tâches. Ces spécifications font l'objet du chapitre III.

1. QUELQUES DEFINITIONS

La *simulation* est une méthode qui consiste, à partir d'un *modèle* d'un système ou d'un phénomène que l'on désire étudier, à faire des expériences et à étudier le comportement du modèle dans différentes circonstances [Pro81].

Or, la définition des concepts sous-jacents à la notion (aux notions ?) de modèle a suscité et suscite toujours de longs débats.

Le modèle selon ZEIGLER [Zei76] est "une représentation compacte d'un objet réel qui, d'une part, reproduit certains comportements de l'objet réel et, d'autre part, fournit un moyen pour dégager les caractéristiques pertinentes de l'objet".

Une définition plus générique est faite par Minsky [Min65] : "soit deux objets A et B et un observateur C ; l'objet A est dit "modèle de l'objet B" si l'observateur C peut l'utiliser pour répondre à certaines questions concernant l'objet B. L'objet B est souvent appelé système réel".

Un des corollaires que tire Ören [Öre78] de cette définition est que "Le Modèle" d'un système réel n'existe pas mais il existe un ensemble de modèles partiels dont chacun représente un des aspects du système réel.

En effet, le modèle n'est jamais un objet en soi mais il est toujours relationnel [Bac76] : modèle de "un objet" pour "un but" dans "un contexte".

Il n'est en aucun cas une reproduction parfaite du réel mais une reproduction plus ou moins fidèle de certaines propriétés du réel dans un contexte donné. Puisqu'il est impossible de modéliser la réalité dans toute sa complexité, le modèle ne peut que dégrader les données du monde réel qu'il manipule :

° les données "en entrée" sont filtrées par rapport à la problématique posée par l'auteur du modèle (but visé, contexte d'utilisation) ;

° les données "en sortie" sont dégradées par les hypothèses simplificatrices prises au sein du modèle.

Cette étape d'altération du réel est nécessaire pour l'obtention de modèles "calculables" (dont l'utilisation conduit à des résultats pratiques).

Par ailleurs, si la modélisation est la représentation d'un des aspects du comportement dynamique[1] d'un système réel lors de son passage d'un état à un autre, la simulation peut alors être définie comme la technique d'étude expérimentale des processus stochastiques ou déterministes par observation de l'évolution de leurs représentations, généralement dans le temps.

---

1 les modèles "permanents" sont considérés comme un sous-ensemble des modèles au sens large du terme.

2. CLASSIFICATION DES MODELES

Une des causes de la confusion concernant les notions sous-jacentes au concept de modèle est le fait qu'il existe plusieurs classes de modèles. Une des classification des modèles est celle proposée par Ören [Öre78]. Elle est basée sur les cinq critères suivants :

° l'existence (ou non) dans le modèle de variables d'entrée, de variables de sortie et de variables d'état (cf. §2.1) ;

° la relation des différentes variables du modèle avec le temps (cf. §2.2) ;

° la relation entre les différentes variables du modèle (cf. §2.3) ;

° le formalisme du modèle (cf. §2.4) ;

° la composition interne du modèle (cf. §2.5).

2.1. EXISTENCE DE VARIABLES D'ENTREE, DE VARIABLES DE SORTIE ET DE VARIABLES D'ETAT

Un modèle est dit "à mémoire" s'il possède au moins une variable d'état (variable dont une dérivée, généralement la dérivée par rapport au temps, entre dans la formulation du modèle). Dans le cas contraire, le modèle est dit "sans mémoire" (cf. Fig. II.1).

Un modèle est dit "autonome" s'il ne possède aucune variable d'entrée. Il est dit "non-autonome" dans le cas contraire.

Un modèle est dit "fermé" s'il ne possède aucune variable de sortie. Il est dit "ouvert" dans le cas contraire.

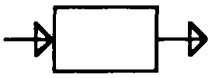
Types de modèles			Variables d'état	Variables d'entrée	Variables de sortie	Représentation iconique
Modèle sans mémoire (fonction instantanée)				X	X	
Modèle à mémoire	Modèle autonome	Modèle fermé	X			
		Modèle ouvert	X		X	
	Modèle non-autonome	Modèle fermé	X	X		
		Modèle ouvert	X	X	X	

Figure II.1. Classification des modèles basée suivant le critère de l'existence de variables d'entrée, de variables de sortie et de variables d'état.

2.2. RELATIONS DES VARIABLES DU MODELE AVEC LE TEMPS

Suivant la représentation du temps dans le modèle, le modèle est dit discret, continu ou mixte :

° les modèles discrets traitent le temps en succession d'instants particuliers au cours desquels les modèles changent d'états. Les variables d'état et les sorties du modèle ne peuvent être connues que pour ces instants. Dans ce cas, il n'existe pas d'algorithme qui calcule la solution mais une simulation effective du déroulement des séquences temporelles. Cette approche concerne les systèmes présentants des files d'attentes et des problèmes de priorités (simulation des procédés de fabrication, de réseaux de communication...). Ces systèmes sont généralement caractérisés par un certain nombre de grandeurs aléatoires (risque de panne d'une machine, délais de fabrication...).

° les modèles continus sont dirigés par l'évolution dans le temps de phénomènes générateurs de comportement. Le temps est une durée qui intervient souvent d'une façon implicite par l'intermédiaire des équations différentielles (il peut prendre comme valeur toutes les valeurs réelles qui appartiennent à l'intervalle de simulation). Le paradoxe est que, pour résoudre ces équations, une discrétisation du temps est souvent nécessaire. Néanmoins, les variables d'état et

les sorties du modèle peuvent, en principe et indépendamment de la méthode de résolution numérique, être calculées à tout moment .

Les modèles continus (rappelons le, modèles où le temps est considéré comme une durée) peuvent présenter des discontinuités (ces discontinuités sont relatives aux variables et non pas au temps). On peut distinguer deux types de discontinuités :

- discontinuités aux dérivées : dans ce cas, la valeur de la dérivée d'une variable présente une discontinuité (par contre, la valeur de la variable peut être continue - par exemple à l'instant t1 de la Fig.II.2) ;

- discontinuités des valeurs (par exemple à l'instant t2 de la Fig.II.2.).

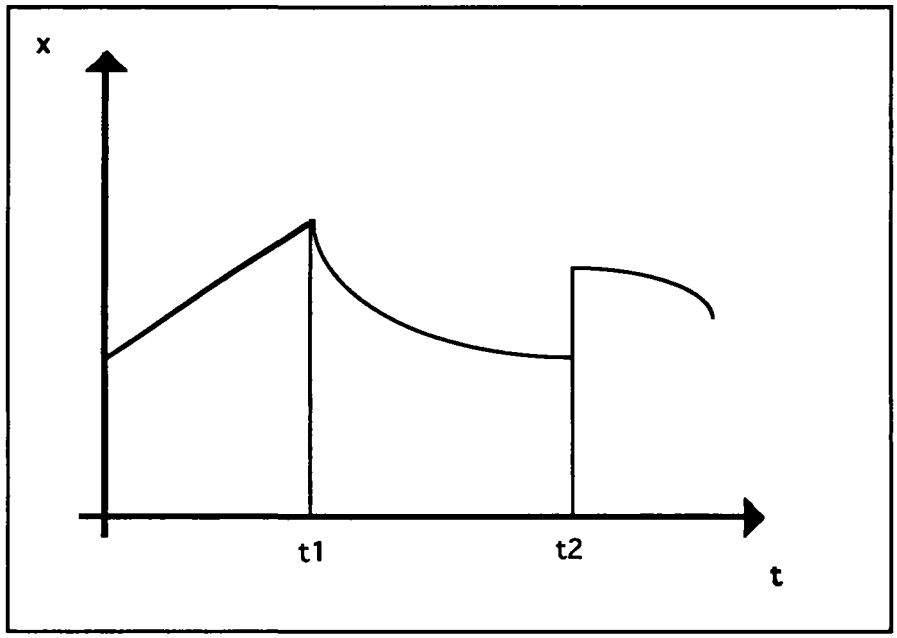


Figure II.2. Discontinuités aux dérivées et discontinuités des valeurs d'une variable.

° les modèles mixtes sont des modèles qui combinent les deux notions de temps : c'est l'union d'intervalles où le temps est décrit comme une durée et d'intervalles où le temps est décrit comme une suite d'instants. Afin que ces modèles mixtes deviennent "calculables", deux approches sont possibles :

- par insertion dans des modèles discrets de blocs d'équations représentant la partie continue ;

- par insertion dans des modèles continus de blocs "automates" décrivant des événements discrets [Nak86].

2.3. RELATIONS ENTRE LES VARIABLES DU MODELE

Les relations entre les variables d'un modèle peuvent être, d'une part, déterministes ou stochastiques (dans ce cas on parle de modèle déterministe ou de modèle stochastique) et, d'autre part, linéaires ou non-linéaires :

° un modèle déterministe est un modèle dont l'état actuel et les entrées déterminent, d'une façon unique, les valeurs futures des variables d'état et de sortie ;

° un modèle stochastique (probabiliste) possède au moins une variable aléatoire et, de ce fait, les valeurs futures des variables d'état et de sortie du modèles ne peuvent pas être calculées d'une façon unique ;

° un modèle linéaire est un modèle dont les variables sont liées par des relations linéaires (relations qui peuvent subir des transformations linéaires) :

soit $y_1(t)$ et $y_2(t)$ les réponses d'un modèle aux entrées $v_1(t)$ et $v_2(t)$,

le modèle est dit linéaire si sa réponse $y(t)$ à une entrée $v(t) = c_1\, v_1(t) + c_2\, v_2(t)$
peut s'écrire sous la forme $y(t) = c_1\, y_1(t) + c_2\, y_2(t)$.

Dans le cas contraire, le modèle est dit non-linéaire.

Le couplage entre modèles linéaires et modèles non-linéaires est un champ d'applications important des problèmes "rigides" (stiff problems).

2.4. FORMALISME DU MODELE

Le choix d'un formalisme pour un modèle se fait relativement à la structure du modèle (cf. Fig. II.3) : certains formalismes sont mieux adaptés aux modèles continus (par exemple, les équations aux dérivées partielles, les équations différentielles...) ; d'autres sont plutôt adaptés à des modèles discrets (équations aux différences finies...).

Les équations aux différences sont l'équivalent discret des équations différentielles : les variables de temps et d'espace y sont discrétisées.

Les machines à états finis peuvent être des machines dépendantes ou indépendantes des états précédents. Un exemple d'application sont les fonctions de transfert discrétisées.

Rapport des variables avec le temps	Evolution des variables	Formalisme
Modèles continus	Variables continues	Equations aux dérivées partielles
		Equations différentielles
	Variables discrétisées	Interaction de processus
		Modèle continu par morceaux
Modèles discrets	Variables discrétisées	Découpage temporel
		Equations aux différences
		Machine à états finis
		Graphes de Markov

Figure II.3. Formalisme des modèles.

Le découpage temporel ("time-slicing") est un formalisme où le temps est incrémenté à des intervalles constants. Après chaque incrémentation, des tests permettent de définir les actions à entreprendre.

Les graphes de Markov sont des modèles autonomes à mémoire dont la probabilité de transition d'un état à un autre ne dépend pas de la manière dont cet état a été atteint (historique de l'état). Le système est présenté sous forme de graphe où chaque nœud correspond à un état. Le passage d'un état à un autre est donné en fonction de la probabilité d'occurrence. Les graphes de Markov peuvent être utilisés séparément ou en relation avec d'autres formalismes. Dans ce cas, il faut définir les valeurs des sorties pour chaque état du système.

Des travaux tels que les graphes de liaison ("bond graph") [Lor87] ou le Formalisme d'Evolution par Transfert (FET[2] - [Lah89]) proposent des représentations à large vocation permettant de résoudre les incompatibilités des formalismes.

2.5. COMPOSITION INTERNE DU MODELE

Un modèle peut être considéré comme :

° un ensemble dont les éléments ne peuvent pas être considérés séparément (dans ce cas, le modèle est dit monolithique) ;

° un assemblage de modèles plus élémentaires dont chacun représente un des aspects du modèle global (un modèle élémentaire peut aussi être décomposé en éléments plus élémentaires encore).

La décomposition récurrente d'un système complexe en éléments plus simples possède les avantages suivants (cf. Fig.II.4.) :

° elle permet de contourner des difficultés de résolution numérique (par exemple, dans le cas d'un système d'équations algébro-différentielles fortement couplées et non linéaires en régime instationnaire) ;

° elle conduit au concept de bibliothèque de modèles. En effet, créés pour des besoins spécifiques, les modèles élémentaires sont facilement ré-utilisables dans des contextes différents sous réserve du respect d'un certain nombre d'hypothèses d'ordre physique, mathématique, et/ou numérique.

---

2 le Formalisme d'Evolution par Transfert développé au LESETH considère un découpage par cellules et le couplage entre les évolutions de ces cellules. Chaque cellule est modélisée par ses variables d'état sous forme d'équations de bilan de la Thermodynamique des Phénomènes Irréversibles.

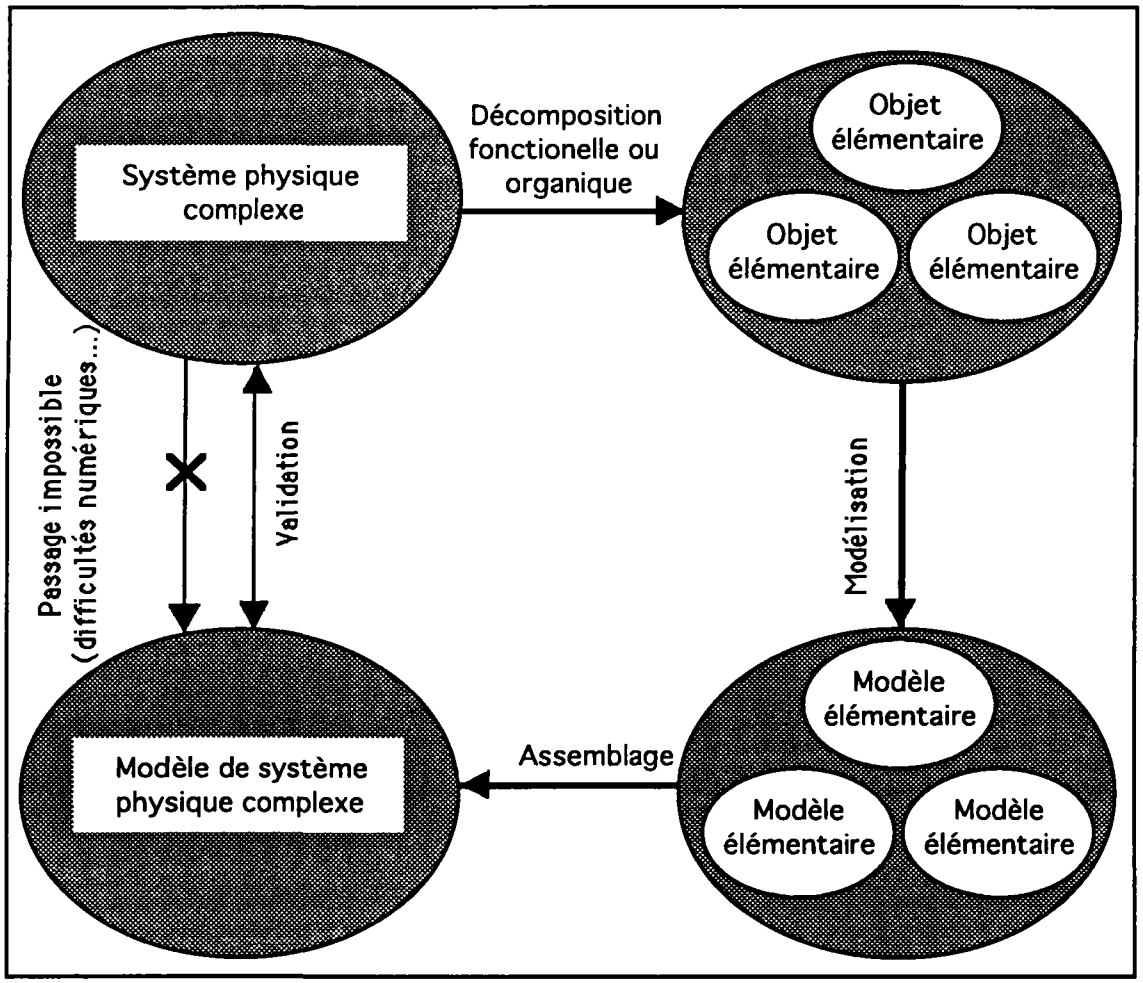


Figure II.4. Processus de décomposition d'un système complexe en objets élémentaires pour contourner des difficultés de résolution numérique.

2.6. DEFINITION DES MODELES UTILISEE DANS L'ESI

Sur la base de la classification présentée ci-avant (cf. §2), les modèles dont il s'agit dans la suite du document peuvent être définis comme l'ensemble des équations algébro-différentielles (implicites ou explicites, linéaires ou non), dont la résolution pour des valeurs initiales et/ou des sollicitations données, fournit une simulation du système. Cette résolution est menée en général par des moyens numériques du fait de la complexité des problèmes qui rend impossible la résolution formelle.

De plus, nous nous intéresserons essentiellement aux cas où la modélisation par assemblage de modèles élémentaires est possible.

3. LE PROCESSUS DE MODELISATION / SIMULATION / ANALYSE DES RESULTATS

A chaque étape du processus de modélisation / simulation / analyse des résultats, la notion de modèle change de caractéristiques : nous distinguerons alors les notions de modèle physique, modèle mathématique, modèle numérique et modèle informatique.

Comme tout processus de conception, le processus de modélisation / simulation est un processus complexe comportant de nombreuses itérations et où sont combinées une approche rationnelle descendante ("top-down") et une approche ascendante ("buttom-up") mettant en jeu des solutions déjà connues. Nous pouvons néanmoins distinguer les étapes suivantes :

3.1. SPECIFICATION DU PROBLEME

Cette étape permet d'une part de définir le contexte de l'étude et, d'autre part, de déterminer les objectifs à atteindre. Les problèmes liés à cette étape viennent du fait que la définition du contexte et des objectifs est généralement incomplète ou incertaine puisque l'étude du problème n'est pas encore réalisée. L'évaluation a priori de la pertinence des objectifs relativement aux difficultés de résolution peut donc être erronée. Généralement, cette évaluation est basée sur une analogie avec des cas déjà rencontrés.

3.2. ELABORATION D'UN MODELE CONCEPTUEL

L'élaboration d'un modèle conceptuel est une étape principale du processus à l'issue de laquelle la structure du modèle, la décomposition en sous-modèles et l'organisation des couplages sont réalisés. On parle alors de "modèle physique". Dans cette étape, on peut distinguer les sous-étapes suivantes :

° définition des frontières du système : le choix des frontières entre le système à étudier et le monde extérieure doit faire en sorte que ces deux entités échangent des informations mesurables pour faciliter la validation expérimentale ultérieure du modèle ;

° décomposition du système en objets élémentaires (décomposition fonctionnelle ou décomposition organique) : cette décomposition est guidée par l'utilisation présumée du modèle (par exemple, la décomposition des objets dans un bâtiment destinée à être utilisée par un modèle de calcul des besoins annuels de chauffage ne prendra pas en compte les parois intérieures) ;

° estimation des grandeurs/phénomènes significatifs pour chaque objet : on admet implicitement que les grandeurs ou phénomènes qui ne sont pas déclarés significatifs ne sont pas pris en compte par le modèle ; le danger dans cette étape est de passer à côté de corrélations intéressantes puisqu'elles ne sont pas connues a priori ;

° définition des couplages entre les objets : ces couplages peuvent être topologiques, phénoménologiques ou fonctionnels.

3.3. MODELISATION MATHEMATIQUE

Le "modèle physique" issu de l'étape précédente est traduit par des formules mathématiques. Il perd dans cette traduction une partie de la sémantique qui lui était rattachée puisque ces formules n'ont pas ou presque pas de références aux composants réels qu'ils représentent. On parlera à ce stade de "modèle mathématique". La modélisation mathématique comprend les étapes suivantes :

° écriture des équations "physiques" liants les attributs des entités définies dans la phase précédente. Pour des systèmes déterministes, ces équations sont [Ril86] :

- des équations d'état (équations liants, pour chaque solide ou liquide, la pression, la température, le volume, la composition chimique...) ;
- des fonctions thermodynamiques (énergie interne, enthalpie, entropie...) ;
- des équations de bilans (de masse, d'énergie, de composant chimique, de quantité de mouvement...) ;
- des équations de flux liant des flux à d'autres flux ou des flux à des forces motrices ;
- des équations cinétiques caractérisant les sources et les puits ;
- des équations d'équilibre mécanique, thermique... ;
- des lois de comportement des matériaux (élastique, plastique, visqueux...) ;
- des équations de propagation pour les phénomènes de nature vibratoire (lumière, rayonnement thermique...).

° écriture des conditions initiales, des conditions aux limites et des contraintes ;

° analyse de la structure du système d'équations (caractère différentiel, linéarité/non linéarité, stabilité du système d'équations...).

3.4. CHOIX D'UNE METHODE DE RESOLUTION NUMERIQUE

Dans cette étape, le "modèle mathématique" est associé à une méthode de résolution numérique. Le "modèle numérique" ainsi obtenu devient "calculable".

Puisqu'il n'existe pas en général de solutions analytiques aux modèles complexes, les méthodes de résolution sont des méthodes itératives. Pour les systèmes non-différentiels linéaires, les méthodes utilisées sont du type Gauss, relaxation ou optimisation. Pour les systèmes différentiels linéaires ou linéarisables dépendant d'une variable (le temps en général) les méthodes de résolution utilisent une discrétisation du temps (Euler, Runge-Kutta, inversion de matrice, méthodes de tir en particulier GEAR...). Le choix d'une méthode de résolution adaptée peut apparaître comme une contrainte pratique par rapport au développement du modèle proprement dit. En effet tout ou partie de cette étape peut être automatisée et l'utilisation de techniques spécifiques (calcul formel, identification de forme...) peut y contribuer.

3.5. CODAGE

L'étape de codage représente la mise sous forme informatique du modèle. On parle alors de "modèle informatique" ou de "module". Bien que la plupart des travaux de modélisation se font actuellement en utilisant des langages généraux (Fortran...) ou des langages de simulation (ASCL, CSSL, GPSS...), d'autres voies sont possibles.

En effet si on considère que l'implémentation informatique des modèles mathématiques fait partie de l'activité de développement informatique au sens large, elle doit donc répondre à certaines contraintes de génie logiciel. Deux de ces contraintes sont importantes dans le cadre du développement de modèles : la réutilisabilité et l'extensibilité.

La réutilisabilité de composants logiciels (en l'occurrence des modèles) doit permettre à un usager de réutiliser facilement un composant et/ou de le communiquer à un autre usager. Ceci implique une documentation satisfaisante des composants (l'exemple à suivre dans ce domaine est celui des électroniciens ; un composant électronique, une fois définies les entrées qu'il nécessite, les sorties qu'il fourni et les conditions d'utilisation qui lui sont propres, peut être utilisé à maintes reprises dans des circuits différents).
Suivant l'environnement de développement choisi, l'utilisateur aura plus ou moins d'utilitaires à sa disposition lui facilitant la tâche. Certains des environnements orientés objet (Eiffel...) permettent par exemple de vérifier automatiquement la cohérence des assertions et de générer automatiquement la documentation [Mye88].

L'extensibilité des composants logiciels implique la possibilité d'étendre des procédures déjà existantes à des cas proches au prix de peu de modifications. Par la suite on dira que les composants logiciels doivent être "continus" c'est à dire que des modifications minimes dans les spécifications entraînent des modifications minimes dans la structure du composant mais pas une remise en question de la structure.

Des techniques issues des environnements orientés objets (héritage, généricité ...) permettent de répondre à ces contraintes.

Bien que l'utilisation d'environnements orientés objet pour la modélisation est à présent rare, il n'en demeure pas moins que c'est une technique prometteuse puisqu'elle répond à une des grandes préoccupations du développeur de modèle qui est de faciliter le transfert de la connaissance liée au modèle.

3.6. VALIDATION

Cette étape détermine le degré de confiance accordé au modèle. La validation du modèle composé d'un assemblage de modèles élémentaires passe par une validation séparée des modèles élémentaires suivie d'une validation de l'agrégation de modèles afin de localiser facilement les erreurs. Une ou plusieurs des méthodes suivantes peuvent être utilisées :

- l'analyse de sensibilité du modèle vis à vis de ses variables. A l'issue de cette étape, la précision requise de chaque variable doit être définie. Les méthodes d'organisation de l'analyse de sensibilité relèvent des techniques de "plan d'expérience" ("Experiment Design") [SV82] ;

° la validation "comparative" (par rapport à d'autres modèles) est délicate à mettre en œuvre : l'égalité des entrées mais aussi des hypothèses de base des deux modèles doivent être assurée ;

° validation par simplification ou validation analytique : la résolution analytique directe n'étant généralement pas possible, la validation se fait après simplification du modèle par attribution à certaines variables des valeurs constantes. Les sorties du modèle sont alors comparées aux résultats de la solution analytique qui dans ce cas peut être calculée ;

° validation expérimentale : la validation expérimentale n'est possible que si le modèle et l'expérimentation ont des "structures conceptuelles" semblables pour pouvoir "segmenter" le modèle en parties dont la vérification expérimentale est possible. La définition des écarts entre les valeurs calculées et les valeurs mesurées doit prendre en compte les problèmes de précision de mesure ;

° techniques d'identification : l'identification est un élément central pour les modèles de commande et les modèles stochastiques. Elle permet d'optimiser l'écart entre les valeurs expérimentales et les valeurs calculées en faisant varier certains paramètres du modèle (Fig.II.5).

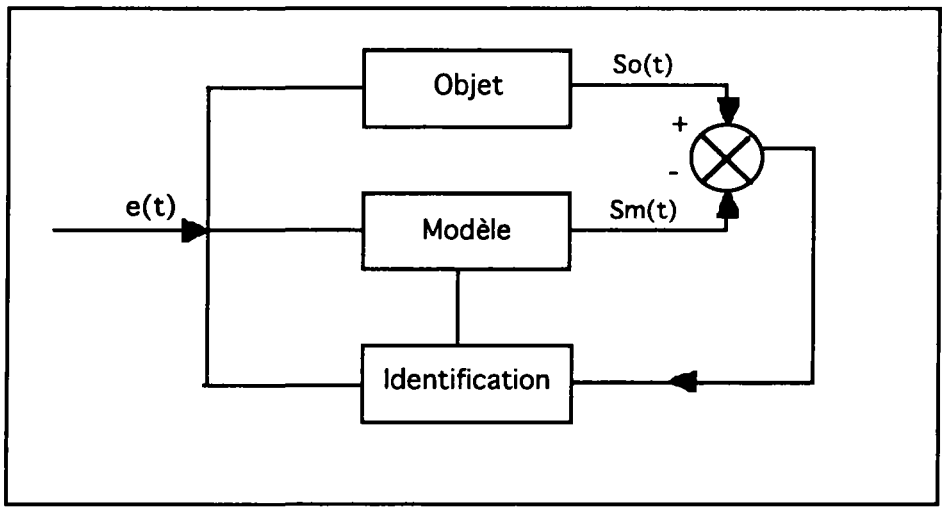


Figure II.5. Techniques d'identification.

3.7. SIMULATION

Cette étape comprend la définition des valeurs de simulation qui permettent d'atteindre les objectifs fixés dans la première étape (étude paramétrique) ainsi que l'exécution de la simulation. Le choix des valeurs pertinentes peut être hasardeux ; il doit être guidé par des données expérimentales.

3.8. ANALYSE DES RESULTATS

L'analyse des résultats se fait en fonction de la limite de validité du modèle mais aussi de la sensibilité de certains paramètres du modèle et de l'estimation des erreurs survenus lors du traitement numérique et du traitement informatique. Les questions qui se posent à ce stade sont :

- dans quel contexte les résultats sont-ils valides ?
- dans quelle mesure peut-on les généraliser ?

L'introduction de la notion d'intervalle de confiance dans la valeur d'une variable peut assurer une certaine sécurité à l'usage du modèle (Fig.II.6).

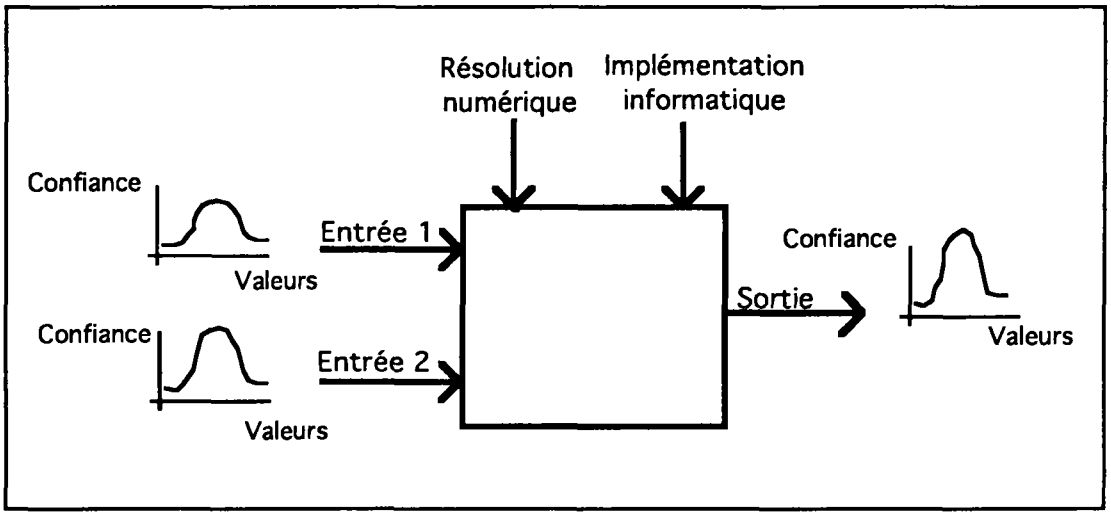


Figure II.6. Intervalle de confiance rattaché aux variables du modèle.

La confiance dans la valeur d'une variable peut être définie comme la probabilité que cette valeur soit valide. Pour les variables calculée, la confiance peut être calculée en fonction de :

° la confiance rattachée à chaque variable d'entrée ;

° les erreurs dues à la méthode de résolution numérique ;

° les erreurs dues à l'implémentation informatique (erreurs de troncation...).

Pour les variables mesurées, la confiance est basée sur l'estimation des erreurs de mesures commises.

3.9. ITERATIONS DURANT LE PROCESSUS

Les étapes du processus de modélisation / simulation / analyse des résultats sont éminemment itératives : chaque étape peut comporter de nombreuses itérations et/ou des retours vers des étapes précédentes (Fig.II.7).

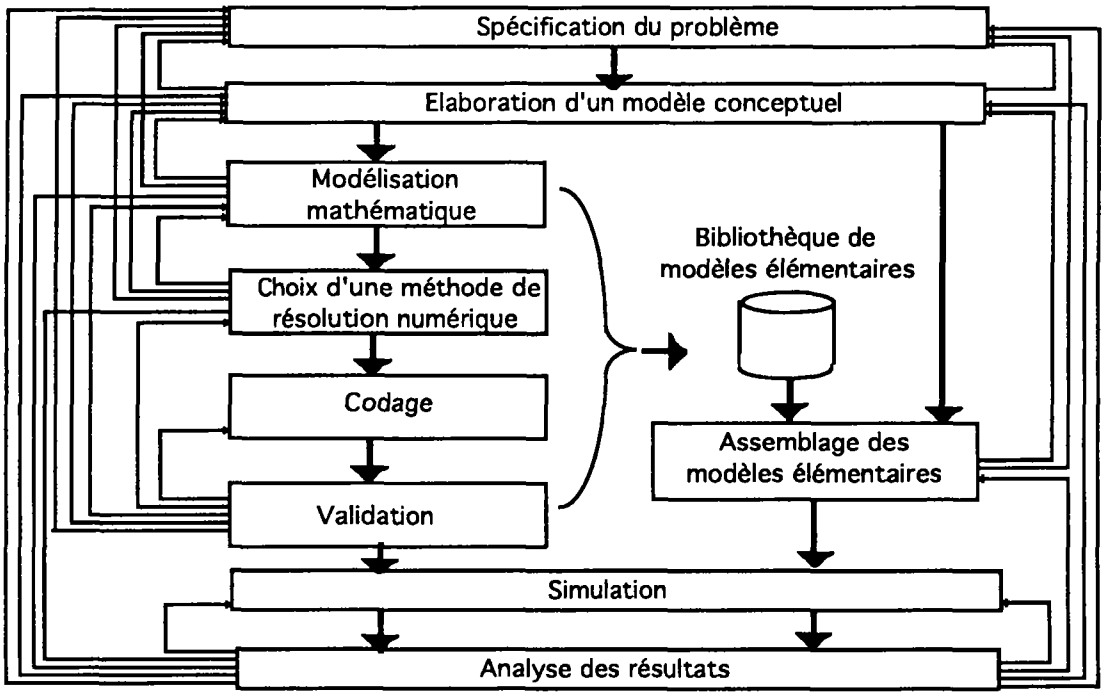


Figure II.7. Processus de modélisation / simulation / analyse des résultats.

La constitution d'une bibliothèque de modèles validés et documentés est une approche qui permet de diminuer les itérations et d'optimiser le processus global [Dub88].

L'existence d'une telle bibliothèque destinée à un ou plusieurs corps de métier du bâtiment est un vecteur important de transfert de la connaissance des chercheurs vers les professionnels et un moyen de valorisation des travaux de recherche.

4. OUTILS DE SIMULATION DU BATIMENT

La simulation du bâtiment est intéressante puisqu'elle permet de remplacer une expérimentation souvent difficile voire impossible. Néanmoins, certaines difficultés sont inhérentes à la simulation du bâtiment. Ces difficultés sont les suivantes :

° l'hétérogénéité des éléments et des phénomènes ;

° les problèmes numériques dus en particulier aux couplages forts entre systèmes à constantes de temps très différentes ;

° les effets incertains et imprévus du comportement des occupants et de la météo ;

° la saisie des données est lourde qu'il s'agisse de la description des projets de bâtiments, des données de l'environnement ou des données constructeurs dont la normalisation est loin d'être achevée ;

° les difficultés spécifiques à l'interprétation des résultats qui nécessitent l'identification correcte de leurs domaines de validité.

La plupart des outils informatiques utilisés actuellement dans les métiers relatifs à la thermique du bâtiment ont été développés durant les quinze dernières années [Sow91]. Ces outils traitent essentiellement les aspects suivants :

° calcul des besoins en chauffage et en charge frigorifique ;

° dimensionnement des équipements ;

° définition des conditions de confort ;

° vérification de la conformité réglementaire ;

° conception architecturale ;

° codes de simulation. Ces codes de simulation peuvent être séparés en deux classes distinctes :

- les codes modulaires détaillés (TRNSYS[3] , HVACSIM+[4]...) ;

- les codes monolithiques qui traitent le bâtiment dans sa globalité (DOE[5]...).

Ces différents logiciels sont, pour la plupart, issus de langages et techniques utilisés dans les années 70 et, de ce fait, manquent de flexibilité (améliorations difficiles...) et de généricité (utilisation d'un même logiciel pour différents types de problèmes). L'utilisation des nouvelles techniques informatiques (Programmation Orientée Objet, Systèmes experts, Programmation par les contraintes...) pour améliorer les performances de ces logiciels est une approche suivie par plusieurs équipes de recherche. Par la suite nous présentons une liste non exhaustive des projets concrets issus de ces travaux de recherche.

° Le projet Energy Kernel System (EKS) développé aux Etats Unis (EKS-US) à l'Université de Californie et au LBL (Lawrence Berkely Laboratory) a pour but de développer des outils performants dédiés aux professionnels du bâtiment [Sow91]. Concrètement, ces outils utiliseront une nouvelle version de DOE (DOE-3) enrichie d'une bibliothèque d'équipements de chauffage, de ventilation et de climatisation et d'un solveur d'équations algébro-différentielles conçu par le Pr. E. SOWEL de l'Université de Californie. Une première implémentation de l'EKS-US est réalisée au LBL dans le cadre du projet SPANK.

  Le projet EKS-US est basé sur une approche modulaire à l'échelle des équations mathématiques : la définition du problème à simuler (en l'occurrence un composant technologique) se fait par assemblage de "composants élémentaires" qui sont en fait des équations algébriques ou différentielles. Ces équations sont indépendantes de la méthode de résolution et les variables n'y sont pas orientées ce qui permet une réutilisabilité maximale des modèles. Le système d'équations ainsi obtenu est réduit avant d'être résolu (le nombre d'inconnues est optimisé afin de réduire les temps de calcul).

° Bien que portant le même nom, le projet EKS développé au Royaume-Uni (EKS-UK) aux universités de Strathclyde et de Newcastle n'a pas les mêmes objectifs [Cla91] : contrairement au projet EKS-US, EKS-UK ne vise pas à développer des applications spécifiques mais un "atelier générique" qui permettra, dans une deuxième phase, d'intégrer des applications spécifiques (ESP, BLAST...). La généricité du projet est basée sur une structuration des données du bâtiment indépendamment des outils : la définition du problème à simuler suit une approche descendante avec des relations du type "est composé de" (par exemple : un problème de simulation du bâtiment est composé d'un site, d'un bâtiment, d'un solveur..., le bâtiment est composé de pièces qui à leur tour sont composés de surfaces, de zones...). La définition d'une hiérarchie figée des composants qui a l'avantage d'être facilement applicable par les professionnels du bâtiment se heurte néanmoins a des problèmes de flexibilité. En

---

3 TRNSYS est un code de simulation développé au Solar Energy Laboratory de l'Université de Wisconsin en 1979.

4 HVACSIM+ est un code de simulation produit par le NBS (National Bureau of Standards) aux Etats-Unis. Il permet une intégration par la méthode de GEAR à pas variable.

5 DOE est un code de simulation développé par le Lawrence Berkeley Laboratory et le Los Alamos Scientific Laboratory en 1979.

effet, il n'est pas possible de définir une telle hiérarchie pour les équipements du bâtiment (ventilateurs, radiateurs...) qui peuvent avoir des relations aléatoires suivant le problème.

° En Belgique, le projet MS1 (Modelling System 1) de F. LORENZ est axé sur la communication entre les différents formalismes : la description des modèles peut se faire dans des formalismes différents ; ces formalismes sont ensuite traduits dans un formalisme normalisé utilisant la théorie des graphes de liaison (bond graph).

° En France, ALLAN a été le fruit d'une collaboration entre Gaz de France et la CISI [PS86]. ALLAN est un pré-processeur graphique pour Neptunix[6] [Nak86] et ASTEC[7]. Les travaux actuels portent sur le développement et la validation d'une bibliothèque de modèles du bâtiment et des équipements associés.

° CLIM 2000 est en cours de développement à Electricité de France [Ron90]. CLIM 2000 est un code de simulation modulaire basé sur une approche nodale dont la particularité réside dans l'utilisation d'une méthode de modélisation prédéfinie associée à un formalisme de documentation des modèles.

Enfin, deux problématiques semblent être partagées par les équipes de chercheurs travaillant sur les "codes de simulation de nouvelle génération" :

° échanges des données : les codes de simulation existants nécessitent la saisie de données relatives au problème posé. Vue la complexité des données caractérisant les bâtiments, cette tâche peut être fastidieuse. Or beaucoup de ces données sont communes à plusieurs domaines : par exemple, le calcul des apports gratuits, nécessite entre autre, la saisie de données spécifiques sur les vitrages (coefficient de déperdition, facteur d'ombre, réflectance...). D'autre part, une étude de la sécurité au feu requiert, pour les vitrages, un autre type de données spécifiques (résistance au feu...) ; néanmoins, il existe des données communes à ces deux domaines (propriétés des matériaux, données géométriques...). La problématique de l'échange des données s'attache à définir les données communes à différents domaines techniques et à les stocker dans des bases de données composants. Ces bases de données pourront ensuite être exploitées par des codes de simulation différents, l'utilisateur se limitant alors à compléter les données spécifiques à son problème.

° bibliothèques de modèles : la création de telles bibliothèques comporte plusieurs avantages [Dub88] :

- faciliter la valorisation des travaux de recherche ;
- économiser le temps nécessaire pour la modélisation ;
- participer à la validation du modèle à la suite d'une large diffusion et utilisation.

---

6 NEPTUNIX est un progiciel de construction de simulateurs pour processus mixtes (continus et discrets) conçu par M. NAKHLE et commercialisé par le CISI.

7 ASTEC est un code de simulation des systèmes suivant une approche par analogie éléctrique développé au CEA et commercialisé par le CISI.

Une telle bibliothèque doit disposer d'une description soignée des équations des modèles mais aussi des informations nécessaires à son utilisation correcte. Ce constat est à l'origine de nombreux travaux sur la documentation des modèles dont les travaux sur le PROFORMA entrepris au GER ALMETH[8] sous l'impulsion du CSTB.

BILAN ET CONCLUSION DU CHAPITRE 2

Comme il a été présenté, la notion de modèle change d'attributs en fonction du plan dans lequel on se place. Quatre plans peuvent être distingués :

° le plan physique où interviennent les principes de base et la décomposition en objets élémentaires couplés ;

° le plan mathématique où chaque élément est associé à un ensemble de relations mathématiques ;

° le plan numérique où la solution de l'ensemble équationnel est apportée par un ensemble de procédures algorithmiques ;

° le plan logiciel où les procédures algorithmiques sont codées.

Chacun de ces plans possède des concepts et des modes de raisonnement qui lui sont propres ; néanmoins ils communiquent entre eux (par exemple, l'événement physique "l'aquastat coupe la chaudière à gaz" se traduit par un modèle avec une discontinuité et, éventuellement, par des problèmes de résolution numérique).

Pour que l'utilisateur puisse avoir un retour d'information pertinent durant toutes les étapes du processus de modélisation / simulation / analyse des résultats, il est indispensable que la communication entre ces différents plans se fasse sans que les informations échangées perdent de leur sémantique. Au sein d'un ESI ceci est réalisé par la combinaison de deux parties :

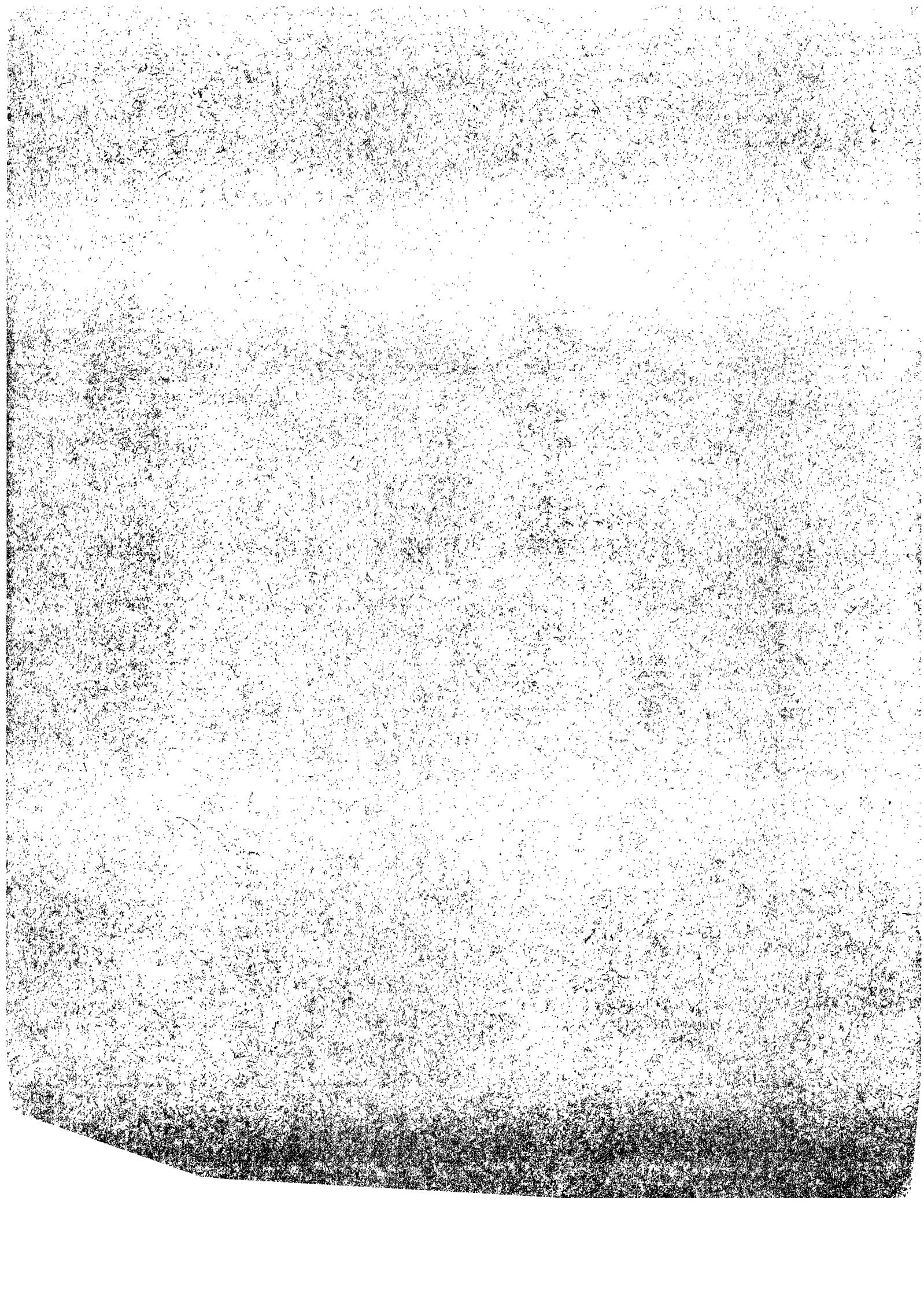
° une partie symbolique qui contient la sémantique des entités manipulées dans un projet de simulation ;

° une partie numérique qui prend en charge la résolution numérique du projet de simulation.

Par ailleurs, la modélisation du bâtiment comporte des difficultés inhérentes au domaine (lourdeur de saisie des données, problèmes numériques...). Des environnements de modélisation du bâtiment d'une nouvelle génération doivent proposer des solutions à ces problèmes. Certains travaux de recherche concernant l'échange des données et la constitution de bibliothèques de modèles commencent à apporter des éléments de réponse.

---

8 GER ALMETH : Groupement d'Etude et de REcherche - Atelier Logiciel pour la Maîtrise de l'Energie dans le Tertiaire et l'Habitat.



CHAPITRE III : SPECIFICATIONS POUR UN ENVIRONNEMENT DE SIMULATION INTELLIGENT

INTRODUCTION

La problématique la plus générale de la réalisation d'un ESI est la suivante :

> En présence d'un système technologique susceptible d'être décrit par une architecture dont chaque élément est identifiable et modélisable, comment concevoir l'assistance que peut apporter une machine à un utilisateur désirant étudier le système considéré au regard de contraintes attachées à ses entrées et sorties ainsi qu'à certaines variables internes ?

L'objectif de ce travail de recherche n'est pas de produire une modélisation d'un système technologique particulier mais de produire un modèle d'un environnement de simulation générique (ESI) : il s'agit d'un environnement logiciel qui permet d'une part d'intégrer des codes de simulation différents et, d'autre part, de faire communiquer ces codes entre eux. Le modèle de l'ESI peut être utilisé dans plusieurs domaines applicatifs par ajout de classes décrivant les systèmes technologiques spécifiques aux domaines. En ce qui concerne le bâtiment, le modèle de l'ESI intègre le travail de modélisation issue du projet *COMBINE* [DEL92].

Dans ce chapitre, nous présentons une modélisation d'un ESI (cf. §3) : il s'agit de définir les données nécessaires au fonctionnement de l'ESI ainsi que les procédures qui manipulent ces données.

Le modélisation proposée de l'ESI doit répondre à la problématique définie ci-avant et respecter certaines exigences sur le plan de la communication H/M (cf. chapitre I) ainsi que certaines exigences découlant des spécificités du processus de la modélisation / simulation / analyse des résultats (cf. Chapitre II).

Pour respecter les exigences relatives aux spécificités du processus de modélisation / simulation / analyse des résultats, et afin de prendre en compte le partage des tâches au sein d'une équipe travaillant dans ce domaine, une étape d'analyse de l'activité a été réalisée (cf. §2).

Le formalisme utilisé pour la modélisation d'un ESI est la Représentation Orientée Objet (ROO). Le choix de ce formalisme est guidé d'une part par les avantages de la ROO sur le plan du génie logiciel en général (cf. §1.2) et, d'autre part, par les avantages spécifiques à l'activité de modélisation (cf. §1.3).
En préalable à la modélisation de l'ESI, il nous a semblé essentiel de rappeler quelques notions relatives à la ROO (cf. §1.1) afin d'éviter toute confusion lors de l'étape de modélisation.

La modélisation proposée pour l'ESI se veut indépendante du domaine technologique : elle doit être applicable à tout domaine où la simulation représente un moyen d'investigation du comportement des systèmes complexes. En fait, pour certains domaines spécifiques, elle risque d'être incomplète. La validation du modèle proposé ne peut se faire que par son utilisation dans des domaines variés. Cette validation peut conduire à un enrichissement du modèle. Une première application au domaine du bâtiment fait l'objet du chapitre IV.

1. FORMALISME UTILISE : LA ROO

Les avantages de la conception et de la réalisation d'un ESI dans un formalisme de ROO peuvent être classées en deux catégories :

° les avantages sur le plan du génie logiciel qui sont communs à toutes les activités de production logicielle (cf. § 1.2) ;

° les avantages spécifiques au domaine de la modélisation (cf. § 1.3).

La définition des notions de la ROO utilisés par la suite fait l'objet du § 1.1. Cette définition est nécessaire étant donné que l'approche orienté objet est un domaine où le vocabulaire est loin d'être normalisé.

1.1. QUELQUES DEFINITIONS

Dans ce paragraphe, la terminologie utilisée lors de la modélisation de l'ESI (cf. §3) est définie. Nous n'entrerons pas ici dans la distinction entre les différentes alternatives d'implémentation (schéma, frame, prototype...) : les concepts et les mécanismes seront décrits tels qu'ils sont utilisés dans l'étape de modélisation et indépendamment des spécificités de certaines implémentations.

° Objet : entité cohérente et autonome qui représente des éléments du monde réel ou des abstractions. Un objet est constitué d'une partie déclarative (attributs) et d'une partie procédurale (méthodes). Un objet peut être considéré comme un serveur capable d'effectuer des requêtes sur sa propre structure, d'en masquer une partie ou d'en permettre l'accès contrôlé. La notion d'objet telle qu'elle est utilisée ci-après englobe aussi bien les classes que les instances de classes.

Exemple : l'objet "fenêtre".

° Attributs : les attributs (ou champs, variables, slots...) décrivent les objets en explicitant leurs caractéristiques. A chaque attribut peut être associée une valeur. L'ensemble des attributs d'un objet constitue l'état de l'objet.
Des compléments d'information (facettes) peuvent de plus être associés à chaque attribut.

Exemple : les attributs de l'objet "fenêtre" peuvent être : largeur, hauteur, type de vitrage, type de fermeture...

° Méthodes : les méthodes (ou scripts...) sont la description du comportement d'un objet face à certaines actions.
Les méthodes d'un objet sont "encapsulées" : elles ne sont connues par les autres objets que par l'intermédiaire d'une interface de communication.
Les méthodes peuvent être déclenchées par l'objet lui-même ou par d'autres objets par envoi de messages.

Exemple : les méthodes de l'objet "fenêtre" peuvent être le calcul de la surface vitrée, le calcul du facteur solaire de la paroi vitrée...

° Facettes : les facettes permettent de décrire les caractéristiques intrinsèques des attributs. Elles peuvent, par exemple, définir des filtres sur l'attribution des valeurs (associer des contraintes sur les valeurs, spécifier le type de la valeur...), des attachements procéduraux...

Exemple : A l'attribut "type de fermeture" de l'objet "fenêtre", on peut associer une facette décrivant la liste des valeurs possibles de cet attribut : {aucune, volets battants, volets roulants...}.

Dans un environnement donné, certaines facettes peuvent être prédéfinies. Dans le modèle de l'ESI proposé, les attributs utilisent les facettes suivantes :

* la cardinalité ; elle permet de spécifier si un attribut A peut prendre une valeur unique ou pas.

  - Les attributs monovalués sont ceux qui, pour une instance donnée et à un instant donné, ne peut avoir qu'une valeur et une seule (exemple : la surface d'une paroi) ;

  - Les attributs multivalués sont ceux qui, pour une instance donnée et à un instant donné, peuvent prendre plusieurs valeurs (exemple : l'attribut "connexions possibles" de la classe "variable d'entrée") ;

* les démons ; les démons permettent de définir des attachements procéduraux à une classe ou à un attribut. Ces procédures peuvent être déclenchées lors de l'accès à l'attribut (lecture, écriture) ou dans des conditions prédéfinies (instanciation, suppression...) ;

* type ; la notion de type permet de spécifier les valeurs qu'un attribut A peut prendre. Elle permet une vérification par le système de la validité des valeurs affectées à l'attribut.
Des exemples de types sont entier, réel, objet... Certains systèmes permettent d'éviter les procédures de vérification en affectant à l'attribut le type "quelconque".

  Par exemple, si le type de A est "entier" et que, pour une occurrence donnée, la valeur attribuée à A est un réel, la reconnaissance par le système du type de A (ici un entier) permet de vérifier l'admissibilité de la valeur attribuée (dans cet exemple, la valeur attribuée à A ne serait pas admise).

* facette spécifique ; cette facette est ajoutée aux trois facettes définies ci-avant ; elle est rattachée à tous les attributs de toutes les classes et elle est, a priori, vide. Elle est prévue pour permettre de stocker, pour certaines classes, des informations spécifiques.

° Envoi de messages : la communication entre les objets est réalisée par envoi de messages qui se déclenche sur un objet particulier ou, d'une façon générale, sur une hiérarchie d'objets. Dans ce cas, le premier objet de la hiérarchie qui reconnaît le message l'intercepte et le traite (déclenche la méthode associée au message). Suivant la méthode déclenchée, le message peut s'arrêter à ce niveau ou être propagé, par l'objet récepteur, à d'autres objets de la hiérarchie.

° Classe : il est possible de regrouper les méthodes et les attributs communs à un ensemble d'objets dans une "classe" afin de structurer la connaissance et d'éviter les redondances. Une classe peut être considérée comme un moule à partir duquel on fabrique des exemplaires (instances). Une classe peut avoir des sous-classe.

° Sous-classe : une sous-classe partage la même structure et/ou les mêmes méthodes que la classe (à l'aide du mécanisme d'héritage) et possède, en plus, des attributs et/ou des méthodes qui lui sont spécifiques. Les relations entre une classe et ses sous-classes sont du type "généralisation/spécification".

Exemple : la classe "fenêtre" possède comme sous-classe "fenêtre PVC"...

° Généralisation : est la démarche qui consiste à regrouper, au sein d'une classe mère, les attributs communs à plusieurs classes.

° Spécialisation : démarche inverse de la généralisation. Elle consiste, en partant d'une classe générale, à définir des sous-classes ayant des spécificités par rapport à la classe générale et, par conséquent, des attributs spécifiques en supplément des attributs communs à la classe générale. Par ailleurs, la spécialisation peut concerner les attributs : il s'agit de redéfinir, au niveau d'une sous-classe, les caractéristiques d'un attribut héritées d'une classe mère.

° Instances : les instances d'une classe sont considérées comme des copies de la classe à laquelle ils appartiennent ; elles partagent la même structure et les mêmes méthodes (définies au niveau de la classe) et ne se distinguent que par la valeur de leurs attributs.

Exemple : l'instance "fenêtre en PVC de la cuisine" d'un projet particulier ; largeur = 1,2m et hauteur = 1m.

° Classe abstraite : une classe abstraite (ou virtuelle) est une classe qui représente un concept abstrait et, par conséquent, ne peut pas servir à créer des instances. L'existence de la classe abstraite ne se justifie que pour le regroupement des méthodes et des attributs hérités par ses sous-classes.

Exemple : la classe "élément de façade".

° Hiérarchie : les hiérarchies permettent d'organiser de façon conceptuelle les classes en les regroupant du plus général (classe) au plus spécifique (sous-classe). Les hiérarchies sont à la base du mécanisme d'héritage.

Exemple : la classe "fenêtre" appartient à la hiérarchie "ouverture".

° Héritage : l'héritage a pour but de faciliter le partage d'informations entre les objets. Il peut concerner la structure (les attributs de la classe sont hérités par les sous-classes) et/ou les méthodes (les méthodes de la classe sont héritées par les sous-classes).

L'héritage peut être simple (une classe ne peut avoir plus d'une classe mère). Dans ce cas, les hiérarchies des classes représentent des arborescences avec des relations de généralisation/spécification entre les classes et les sous-classes.

L'héritage peut aussi être multiple (une classe peut hériter de plusieurs classes mères). Dans ce cas, les hiérarchies représentent des graphes orientés acycliques.
Exemple d'héritage simple : la classe "fenêtre" va hériter de l'attribut "type de vitrage" de la classe-mère "paroi vitrée".

Exemple d'héritage multiple : la classe "fenêtre" va hériter d'une part de l'attribut "type de vitrage" de la classe-mère "paroi vitrée" et, d'autre part, de l'attribut "coefficient de pression" de la classe mère "élément de façade".

° Vue : des vues peuvent être rattachées aux hiérarchies des classes. Cette notion permet d'extraire d'une même hiérarchie les classes et les attributs spécifiques à un point de vue donné.

Exemple : on peut extraire d'une hiérarchie de composants du bâtiment une vue "thermique", une vue "acoustique"...

° Méta-classe : une méta-classe (ou classe de classes) est une classe qui implante la structure et les comportements nécessaires à la description des classes. Ceci permet de redéfinir facilement certains aspects des classes (par exemple, redéfinition de la méthode d'instanciation, de la méthode d'initialisation des instances...).

° Méta-objet : la notion de méta-objet permet de séparer la structure de l'entité décrite (définie au niveau de l'objet) de l'implémentation de cette structure (définie au niveau du méta-objet) : à chaque objet (classes, sous-classes, instances) est associé un méta-objet qui contient les informations sur l'implémentation de l'objet dans l'environnement hôte ; on parle alors de réflexivité du système. La réflexivité d'un système permet de ré-implanter le système dans un autre environnement (par une redéfinition des méta-objets) sans modifier la structure des objets existants.

1.2. AVANTAGES GENERAUX DE LA ROO

Dans ce paragraphe, nous exposons les avantages de l'utilisation de la ROO dans la conception et la réalisation de produits logiciels en général : ces avantages bénéficient, entre autre, à la conception et la réalisation d'un ESI.

1.2.1. Modélisation de la connaissance

Durant l'étape de conception d'un outil informatique, une modélisation du monde d'investigation est nécessaire.

En effet, les outils informatiques sont utilisés pour répondre à certaines questions (par exemple, les outils de simulation...), pour interagir avec le monde extérieur (par exemple, les outils pour le contrôle de processus...) ou pour créer de nouvelles entités (par exemple, les outils de PAO...). Les outils informatiques doivent donc être basés sur une modélisation du monde d'investigation, modélisation qui doit être pertinente pour le domaine d'application de l'outil.

Un outil informatique adapté peut alors être considéré comme un modèle opérationnel : modèle, puisqu'il est censé reproduire certains aspects du monde et opérationnel, puisque, pour ces aspects, il doit fournir des résultats pratiques.

Quand le développement d'outils informatiques est considéré comme une production de modèles opérationnels, l'utilisation de la ROO devient "naturelle" : le monde réel étant composé d'objets (parois, fenêtres...), il paraît approprié d'organiser le modèle autour d'une implémentation informatique de ces objets.

1.2.2. Maintenabilité et sécurité

Durant la phase de mise au point, il est aisé de rechercher les parties à corriger : contrairement aux langages classiques où une application est constituée d'un ensemble de procédures séquentielles, les procédures et les données dans un langage objet sont regroupées au niveau d'un composant identifiable ce qui permet d'optimiser les recherches. Par ailleurs, les corrections sont optimisées puisqu'elles ne portent que sur un composant et ne peuvent pas avoir d'effet sur d'autres composants (l'interaction entre les composants ne se fait qu'à travers l'interface de communication).

L'utilisation d'envoi de messages pour la communication entre les objets augmente la sécurité du code : les messages ne peuvent être interprétés que par les objets destinataires (les objets dont l'interface de communication avec l'extérieur reconnaît le message).

1.2.3. Extensibilité

La définition par classes successives liées par héritage renforce la possibilité d'extensibilité incrémentale du système : les objets sont définis pas à pas avec de plus en plus de détails. A la racine de l'arbre d'héritage se trouvent les structures et les méthodes les plus générales. A partir de cette racine les classes sont de plus en plus spécialisées. L'extension du système se fait d'une part par ajout de nouvelles classes et, d'autre part, par ajout d'attributs aux classes existantes.

1.2.4. Réutilisabilité

La définition des composants logiciels en termes d' "objets" (entités cohérentes et autonomes) facilite leur réutilisation dans des contextes différents. Un objet est un module élémentaire réunissant un certain nombre de données qui lui sont propres et des procédures qui manipulent ces données. C'est une portion de connaissance pouvant vivre indépendamment de ses congénères et, par conséquent, pouvant être réutilisé facilement dans des applications différentes.

Par ailleurs, les propriétés d'héritage permettent une réutilisation du code au sein d'une application : les méthodes attachées à une classe sont héritées par toutes ses sous-classes ; seul le code relatif aux méthodes spécifiques d'une sous-classe doit être implanté.

1.3. AVANTAGES DE LA ROO DANS LE DOMAINE DE LA MODELISATION

Une des particularités des environnements de simulation est la quasi inexistence de la frontière entre développeur et utilisateur : contrairement à d'autres outils informatiques où les rôles de développeur et d'utilisateur sont bien séparés, pour les outils de simulation tout utilisateur est un développeur potentiel ; un utilisateur qui commence par utiliser les modèles développés par d'autres peut rapidement être amené à développer ses propres modèles et, par conséquent, à faire du développement logiciel. Ceci signifie que les avantages de la ROO, qui dans d'autres domaines profitent essentiellement au développeur, sont intéressants pour tous les acteurs dans le cycle de vie d'un environnement de simulation. Traduits en termes relatifs aux environnements de simulation, où l'entité de base est le modèle, ces avantages sont les suivants :

° la connaissance liée au modèle est encapsulée : l'implémentation interne du modèle est séparée de l'interface de communication avec d'autres modèles. Ceci permet :

  - de tester et de valider le modèle individuellement avant son utilisation dans un assemblage (chaque modèle de composant connaît ses comportements propres et peut afficher ses caractéristiques à tout moment) ;

  - de faciliter l'utilisation du modèle dans des assemblages : la communication entre les modèles se fait à travers les interfaces de communication ; l'implémentation interne du modèle peut être modifiée sans remettre en question les assemblages utilisant le modèle ;

° les propriétés d'héritage assurent la non-redondance des modèles et facilite l'affinage successif des modèles : les propriétés communes sont regroupées au sein d'une classe représentant un modèle générique ; l'ajout d'un modèle plus spécifique consiste alors à définir une sous-classe.

° l'utilisation de la notion de vue sur les hiérarchies des modèles permet de définir des hiérarchies qui reproduisent la décomposition organique d'un système technologique complexe (la prise en compte des phénomènes est alors ramenée au niveau des vues). Cette bijection entre les modèles et les composants du système technologique permet d'améliorer la lisibilité et la réutilisation des modèles.

° l'utilisation de la réflexivité assure aux modèles implantés dans un système une portabilité accrue. Ceci permet d'avoir des modèles dont la durée de vie est indépendante des contraintes matérielles et logicielles. L'évolution des modèles ne dépend alors que de l'évolution du capital de connaissances qu'ils représentent.

2. ANALYSE DE L'ACTIVITE

Une analyse de l'activité des utilisateurs de l'ESI est nécessaire afin de déterminer les exigences de conception. Cette analyse doit couvrir tout le cycle de vie de l'ESI et doit aboutir à :

° une identification des tâches réalisées (spécification des buts poursuivis, identification des procédures, mises en œuvre) ;

° détermination des conditions du bon déroulement des tâches.

L'étape de l'analyse de l'activité dans le processus de conception de l'ESI est primordiale : elle permet d'une part d'assurer l'adéquation de l'outil aux conditions d'utilisation (minimiser les difficultés, éviter la redondance des tâches) et, d'autre part, de cerner les tâches durant lesquelles l'outil peut offrir une assistance à ses utilisateurs.

2.1. LES CATEGORIES D'UTILISATEURS DE L'ESI

Le cycle de vie de l'ESI est le suivant :

° conception de l'outil générique ;
° enrichissement des bibliothèques de modèles ;
° exploitation.

Ce cycle de vie comprend des tâches de natures très différentes : une classification de ces tâches est nécessaire. Par la suite, on propose une classification basée sur le niveau d'abstraction des tâches relatives au domaine traité : elle va des tâches les plus génériques (indépendantes du domaine) aux tâches les plus spécifiques (fortement liées au domaine). A chacune des classes ainsi définies est associée une catégorie d'utilisateurs : il s'agit de la catégorie d'utilisateurs qui, au sein d'une équipe, prend typiquement en charge la réalisation de ces tâches.

La classification proposée est la suivante (cf. Fig. 3.1) :

° la description, pour le domaine technologique traité, de la structure des entités manipulées et des modalités du dialogue H/M. A ces tâches est rattachée la notion de "concepteur d'application" (CA) ;

° la constitution, pour un domaine technologique donné, de bibliothèques de modèles de composants. A ces tâches est rattachée la notion d' "utilisateur principal" (Up) ;

° le choix dans les bibliothèques, pour un système technologique donné, des modèles de composants, la constitution d'un assemblage et la définition des données de simulation. A ces tâches est rattachée la notion d' "utilisateur intermédiaire" (Ui) ;

° l'exploitation des assemblages définis par l'Ui afin d'effectuer des études paramétriques (analyses de sensibilité, recherches d'optimum...). A ces tâches est rattachée la notion d' "utilisateur final" (Uf).

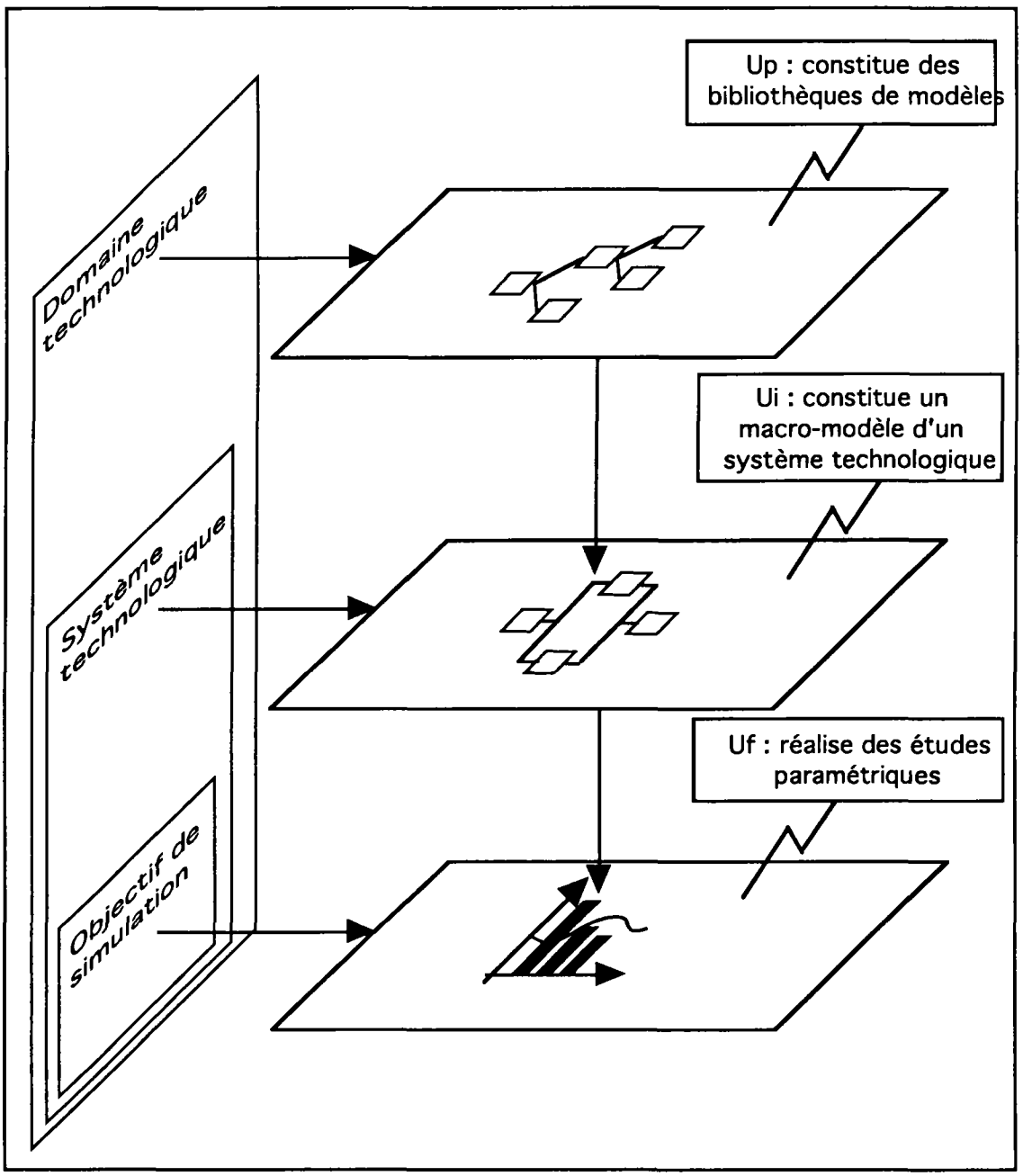


Figure III.1. Définition des tâches par catégories d'utilisateurs.

Par la suite, les tâches associées à chaque catégorie d'utilisateurs seront détaillées (cf. § 2.2 à 2.5). L'étude de ces tâches est ensuite utilisée pour la définition des briques élémentaires composant un ESI (cf. §3).

2.2. CONCEPTEUR D'APPLICATION (CA)

Le CA a la charge de définir d'une part la structure interne de l'ESI (cf. § 2.2.1) et, d'autre part, les modalités du dialogue H/M (cf. § 2.2.1).

2.2.1. Définition de la structure de l'ESI

En se basant sur les classes d'objets prédéfinies au sein de l'ESI (cf. §3), le CA enrichit la connaissance liée à l'ESI en l'adaptant au domaine applicatif traité. La structure de l'ESI perd ainsi de sa généricité mais devient plus adaptée aux spécificités du domaine applicatif traité. L'enrichissement de la connaissance liée à l'ESI se fait soit par un processus de spécification des classes (création de sous-classes) soit par ajout de classes.

2.2.2. Définition des modalités du dialogue H/M

Pour la définition des modalités du dialogue H/M, le CA doit effectuer les tâches suivantes :

° définir les commandes spécifiques à l'application ainsi que leur déroulement. La définition des commandes revient à la programmation des fonctions rattachées. La définition du déroulement des commandes revient à définir, dans une grammaire de type ALG (cf. Chapitre I), la syntaxe des commandes.

  La grammaire définie pour les commandes doit suivre les règles de cohérence et de structuration du dialogue définies dans le modèle de qualité (cf. Chapitre I).

° définir la présentation à l'écran des commandes de l'ESI. Certaines de ces commandes sont indépendantes du domaine technologique (utilitaires graphiques...) d'autres ne le sont pas. Pour les commandes qui dépendent du domaine technologique et afin de réduire les gouffres d'exécution et d'évaluation définis dans la Théorie de l'action (cf. § 1.5 du Chapitre I), le CA doit trouver les icônes qui représentent au mieux les composants technologiques traités ;

° définir la présentation à l'écran des classes de l'ESI qu'il désire rendre utilisables par l'Up, l'Ui et l'Uf.

Les deux dernières tâches sont loin d'être secondaires puisqu'elles permettent de spécifier la "vue externe" de l'ESI qui définit la partie perçue de l'ESI (cf. Fig. 3.2). C'est à travers cette vue externe que les utilisateurs se font une représentation mentale de la partie abstraction des entités et raisonnent sur les systèmes étudiés.

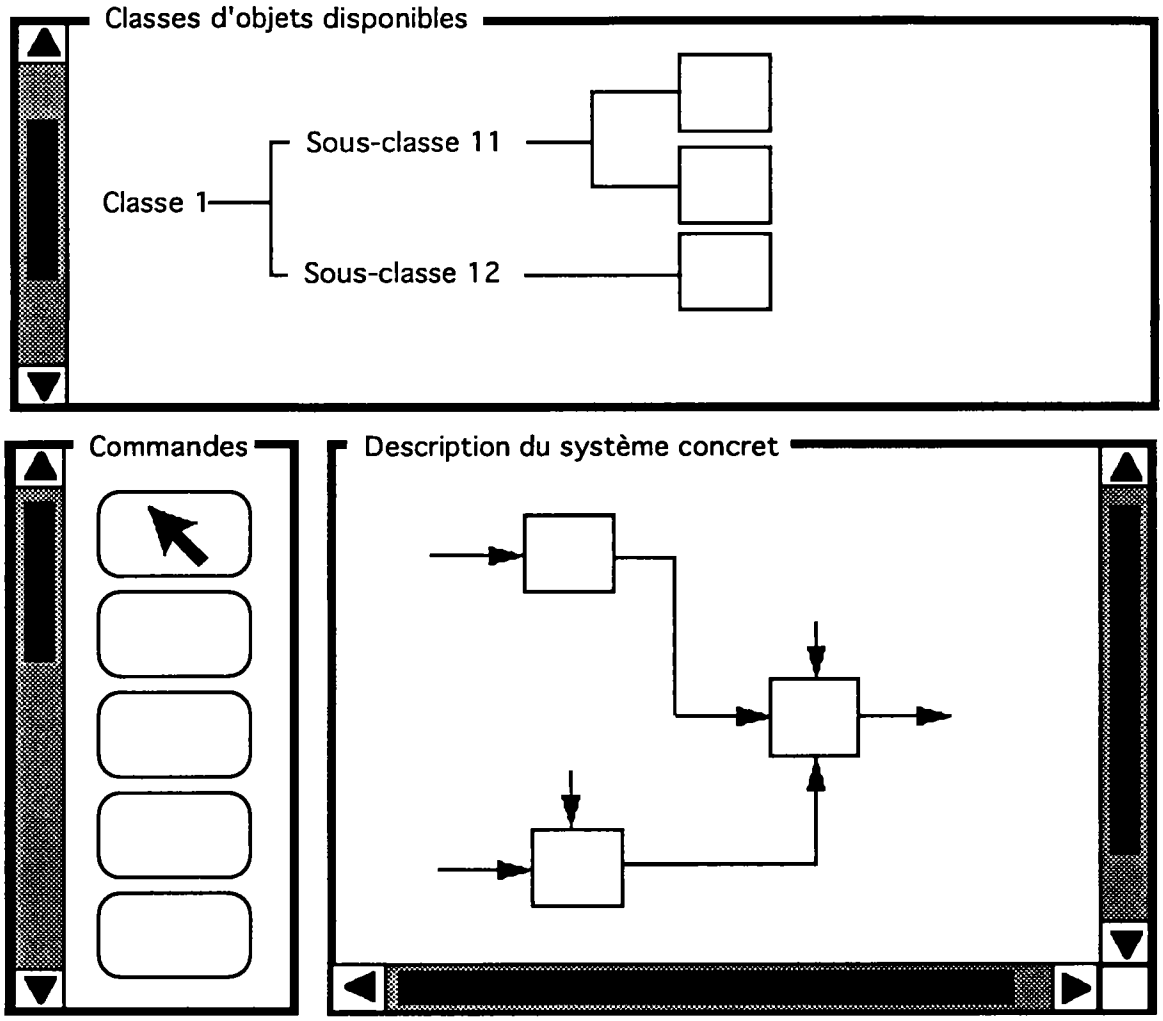


Figure III.2. Exemple de vue externe d'un ESI définie par le concepteur d'application.

Cette méthode de travail du CA basée sur la définition, d'une part, de la sémantique de l'ESI (définie par la structure interne) et, d'autre part, de la syntaxe (définie par la vue externe) permet au CA d'associer à chaque entité graphique un "sens" en décrivant le comportement propre (actions et propriétés associées) et les relations avec d'autres entités graphiques (contraintes d'assemblage...). Le "sens" de l'entité est alors pris en compte par l'ESI. L'entité graphique devient un élément d'un langage *graphique* et *conceptuel* entre l'homme et la machine.

2.3. UTILISATEUR PRINCIPAL (Up)

Il a la charge de créer pour les Ui et les Uf des bibliothèques hiérarchisées de modèles de composants. Ces bibliothèques de modèles seront facilement exploitables à condition de respecter les règles suivantes :

° les modèles qu'elles contiennent répondent bien à la problématique des Ui et des Uf tant sur le plan de la finesse des modèles que sur celui du choix des composants modélisés ;

° les modèles sont hiérarchisés afin de faciliter l'accès aux modèles recherchés ;

° la hiérarchie des modèles est flexible : elle peut être manipulée suivant des vues différentes (les vues peuvent être liées aux phénomènes prépondérants à étudier et/ou au but de la simulation). Il s'agit de répercuter sur la hiérarchie des modèles le choix par l'utilisateur d'une vue spécifique. Ces répercussions peuvent être de deux ordres :

- des modèles spécifiques à la vue choisie sont mis à la disposition des utilisateurs dans la hiérarchie ;

- pour des modèles déjà existants dans la hiérarchie, des attributs spécifiques à la vue choisie deviennent activables.

Par exemple : pour la hiérarchie des modèles "composants d'un local", le choix de la vue "sécurité incendie" induit la mise à disposition des utilisateurs, outre des modèles communs à toutes les vues, des modèles "extincteur", "sortie de secours"...Par ailleurs, le choix de cette vue rend activable l'attribut "mobilité" du modèle "occupant".

° les limites de l'utilisation des modèles sont clairement définies. Ces limites peuvent être inhérentes aux modèles (hypothèses...) comme elles peuvent découler du système technologique étudié (domaine de validité des variables...).

2.4. UTILISATEUR INTERMEDIAIRE (Ui)

En se basant sur un travail de modélisation réalisé en amont, il a la charge de décrire un système technologique existant où en cours de conception.

Suite à la complexité sans cesse croissante des systèmes à étudier, une approche par décomposition récurrente des éléments du système jusqu'à l'obtention d'éléments facilement modélisables semble nécessaire. La décomposition récurrente est basée sur l'hypothèse suivante :

> Tout système technologique complexe peut être défini par un ensemble de composants et un ensemble de relations liant ces composants (la décomposition peut se faire soit sur le plan organique soit sur le plan fonctionnel).

Partant de cette hypothèse, l'Ui utilisera des modèles de composants existants pour créer un conglomérat de modèles (macro-modèle) qui représente certains aspects du système technologique étudié.

Il évident qu'il existe une perte d'information dans le cycle qui consiste à décomposer un système, à modéliser ses composants et à assembler les modèles des composants pour reproduire le comportement du système initial (cf. Chapitre II).

Dans le présent travail, cette perte d'information entre le système initial et l'assemblage de modèles n'est pas prise en compte : ceci signifie qu'on suppose que les étapes de décomposition et de modélisation du système on été réalisées en prenant soin de minimiser la perte d'information.

Le bon déroulement de ces tâches nécessite de la part de l'Ui :

- de pouvoir associer aux composants technologiques (résultant de la décomposition du système technologique) des modèles de composant. Ces modèles doivent être adapté au but de la simulation ainsi qu'aux phénomènes considérés par l'Uf comme intéressants à prendre en compte ;

- d'utiliser de manière adéquate les modèles choisis : il s'agit de pouvoir consulter la documentation afférente qui délimite le cadre d'utilisation des modèles. Cette documentation, rédigée par l'auteur des modèles (Up) spécifie, par exemple, le domaine de validité du modèle et les intervalles de validité de chacune des variables ;

- d'avoir accès à des valeurs par défaut lors de la saisie des données de simulation et des valeurs des variables du modèle ;

- de pouvoir décrire un système technologique suivant des représentations différentes : les informations nécessaires pour décrire un système technologique peuvent être de natures différentes (géométrique, phénoménologique, fonctionnelle...). La méthode la plus adaptée pour la saisie de ces informations change en fonction de leur nature (définition de la géométrie par des modeleurs, définition des phénomènes par des couplages entre schéma-blocs...). L'approche par calque est un moyen pour optimiser le travail de description du système technologique : il s'agit de donner à l'utilisateur la possibilité de renseigner sur chaque "calque" des informations de certaine nature. Les informations ainsi saisies sur un "calque" seront prises en compte par les autres "calques" afin d'en assurer la cohérence.

2.5. UTILISATEUR FINAL (Uf)

L'Uf a la charge d'exploiter les macro-modèles réalisés par l'Ui afin de répondre à un objectif de simulation (dimensionnement d'un composant du système technologique, analyse de sensibilité du système vis a vis d'un de ses paramètre , étude des incidences d'une sollicitation...).

Le bon déroulement de cette tâche nécessite :

° une vérification de la complétude des données de simulation saisies ;

° que l'Uf ait uniquement accès aux paramètres nécessaires pour réaliser son objectif : les paramètres du macro-modèle qui sont utilisables par l'Uf sont les paramètres qui sont jugés par l'Ui comme intéressants à prendre en compte ; le reste des paramètres est invisible à l'Uf. Ceci rend l'utilisation du macro-modèle, pour objectif donné, plus simple ;

° l'existence d'une relation entre les macro-modèles et les systèmes technologiques qu'ils représentent. Cette relation permet à l'Uf d'avoir accès à des informations techniques ou réglementaires (catalogue constructeurs, Avis Techniques, normes...) nécessaires à l'étude du système ;

° une automatisation des tâches de bas niveau, tâches dont l'exécution ne nécessite pas d'information supplémentaire autre que celle déjà saisie (génération des fichiers, choix de l'échelle adéquate pour la visualisation des résultats...).

3. MODELISATION D'UN ESI

D'une façon générale, il n'existe pas de méthode unique pour modéliser les connaissances : l'évaluation d'une solution de modélisation ne peut se faire qu'en fonction des critères d'opérationnalité, de généricité et d'extensibilité de la solution.

Plusieurs solutions de modélisation de l'ESI peuvent être proposées. La solution détaillée par la suite répond néanmoins aux trois critères cités ci-avant. Un exemple d'utilisation de la structure proposée est décrite dans le chapitre IV.

La modélisation proposée de l'ESI est structurée suivant trois étapes (cf. Fig. III.3) :

° l'ESI générique comporte un ensemble de classes prédéfinies. Il s'agit des classes décrivant des concepts utilisés par les environnement de simulation indépendamment du domaine applicatif. L'ESI générique peut alors servir d'environnement d'accueil pour les codes de simulation et des solveurs numériques de natures différentes et traitant de domaines variés ;

° un ESI spécialisé est un ESI générique auquel le concepteur d'application a ajouté des classes spécifiques à un secteur d'activité donné. Ces classes peuvent décrire d'une part des systèmes technologiques particuliers et, d'autre part, les répercussions de certaines spécificités du secteur étudié sur les classes prédéfinies (l'ESI spécialisé peut alors être considéré comme une sous-classe de l'ESI générique). Par exemple, un ESI générique intégrant une modélisation du bâtiment et de ses équipements est considéré comme un ESI spécialisé. L'ESI spécialisé ne peut pas encore être considéré comme une application opérationnelle puisqu'il n'inclue pas des bibliothèques de modèles reproduisant le comportement des systèmes technologiques décrits ;

° les applications opérationnelles sont des ESI spécialisés que les Up ont enrichi avec des bibliothèques de modèles (les applications opérationnelles peuvent alors être considérées comme des instances d'un ESI spécialisé). Ces modèles décrivent, sous forme de systèmes d'équations algébro-différentielles, le comportement des systèmes et/ou des composants décrits dans l'ESI spécialisé. Par ailleurs, les applications opérationnelles doivent intégrer des codes de simulation permettant la résolutions numériques des systèmes d'équations. Au cas où des codes de simulations existants sont utilisés, des traducteurs assurent la communication entre la partie symbolique (décrites sous forme de classe) et la partie numérique (codes de simulation) des applications. Une autre possibilité consiste, sans utiliser des codes de simulation existants, à encapsuler la partie numérique sous forme de méthodes associées aux classes décrivant les modèles (cf. § 3.2.11).
Les applications opérationnelles peuvent utiliser simultanément des codes de simulation différents. Ces codes échangent alors des données (par exemple, des vecteurs de variables résultats) par l'intermédiaire de la partie symbolique qui pilote la simulation et teste la convergence (cf. § 3.2.17).

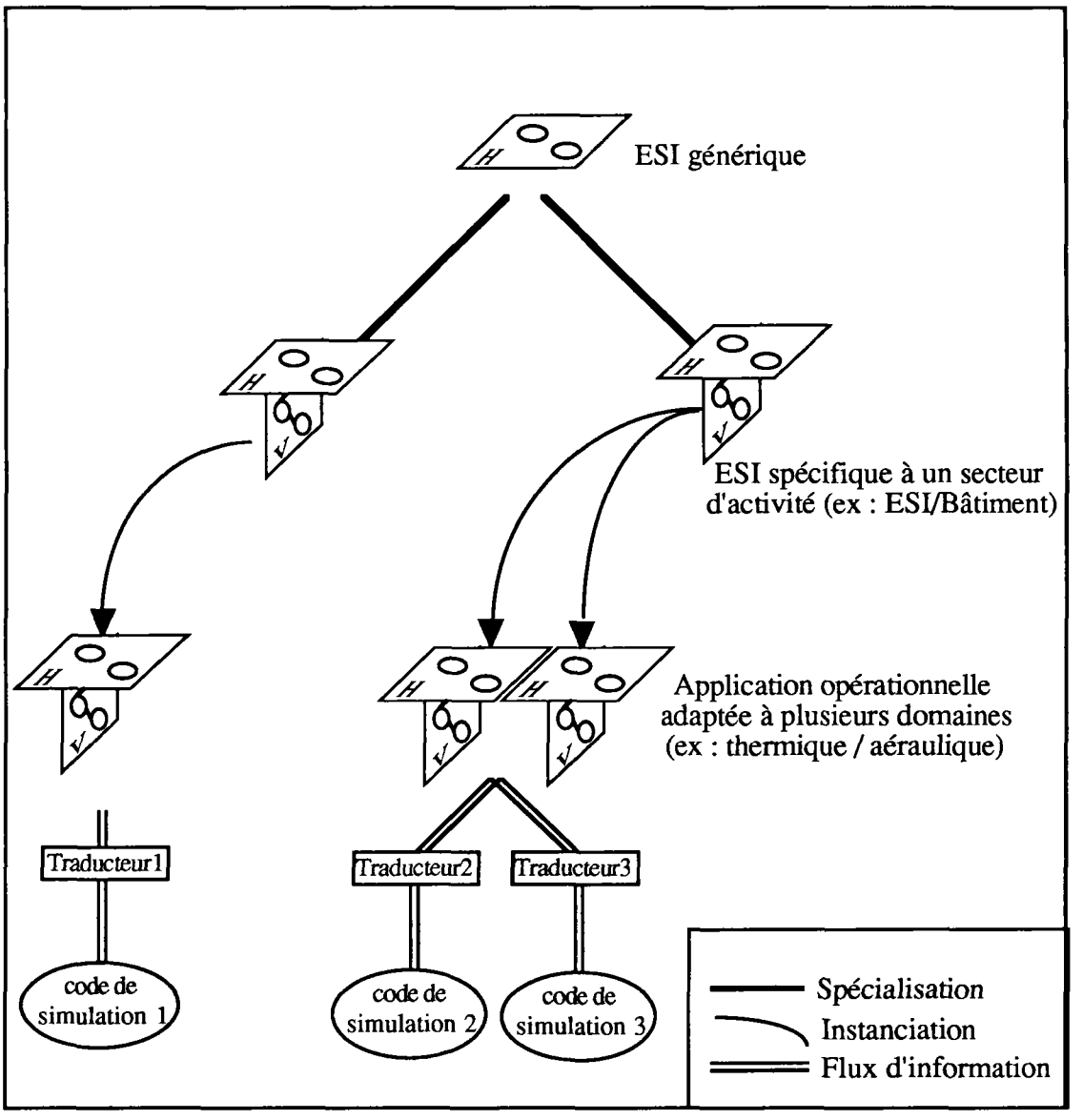


Figure III.3. Relations entre l'ESI générique, les ESI spécifiques et les applications opérationnelles.

3.1. METHODE DE SPECIFICATION : LE MODELE FORMEL HBDS

La spécification d'un ESI doit prendre en compte la couche orientée objet de l'ESI générique à partir de laquelle seront décrites les primitives nécessaires au développement des applications opérationnelles.

Les méthodes de spécification traditionnelles basée sur le cycle d'analyse et de conception (générale et détaillée) se sont avérées inadaptées. Des méthodes basées sur une approche orientée objet ont été considérées [Boo88].

Notre choix s'est porté sur le modèle HBDS (Hypergraph Based Data System) dont l'avantage est de faciliter la transcription de la modélisation formelle en une forme directement utilisable par un système informatique.

3.1.1. Le modèle HBDS

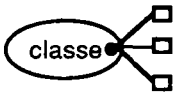
HBDS est une méthode générale permettant de proposer une représentation formelle de structures de connaissances. Le modèle HBDS repose sur un ensemble d'opérateurs (attributs, classes, relations).

Un attribut permet de décrire une entité indépendamment de son environnement. Une classe est obtenue par association des attributs. Des liens sémantiques peuvent exister entre les classes. Une hyper-classe regroupe des classes. Les liens entre ces classes et l'hyper-classe sont de type "généralisation/spécification". Des attributs peuvent être rattachés à l'hyper-classe et/ou aux classes qui lui appartiennent.

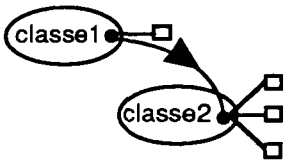
3.1.2. Schématique HBDS

Cette schématique est celle qui est utilisée dans la suite de ce chapitre.

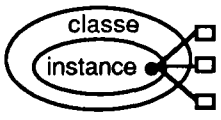
Une classe est représentée par un ovale traduisant l'existence d'un ensemble d'entités observables par le même jeu d'attributs.



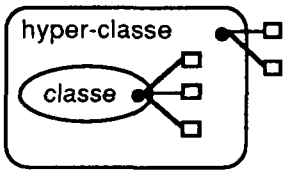
Les attributs sont rattachés en un même point à la classe traduisant l'existence simultanée de ces attributs pour chaque instance de la classe.



Une relation est représentée par un trait orienté entre deux classes.



Une instance est représentée par une double ovale qui contient le nom de l'instance et le nom de la classe à laquelle elle appartient.



Une hyper-classe est représentée par un carré aux coins arrondis qui englobe les classes contenues dans l'hyper-classe. Des attributs peuvent être rattachés à l'hyper-classe.

3.2. PLATEFORME LOGICIELLE : LE SYSTEME MIPS

MIPS, pour "Many Integrated Paradigm System" est un générateur d'applications intégrées multi-expertes, multimédia et hypertextes, développé sur une couche objet à partir de laquelle ont été définies tout un ensemble de primitives : primitives de définition d'objets, outils pour la création et le contrôle de moteurs d'inférence, outils de contrôle de la multi-expertise, module de visualisation 3D...[PBD92]. MIPS est développé au sein de la Division Bases de Connaissances du Service Informatique et Bâtiment du CSTB.

MIPS est utilisé comme plate-forme logicielle pour l'implémentation et la validation du modèle de données de l'ESI. Ceci permet d'une part de tirer profit des primitives de définition d'objets et, d'autre part, d'intégrer à terme au sein de l'ESI des fonctions de raisonnement.

Les relations entre les classes ne possèdent pas une méthode d'implémentation explicite : une relation entre deux classes est décrite par l'affectation à un attribut de la première classe une instance (ou liste d'instances) de la seconde classe. Par exemple la relation "est-composé-de" entre la classe "Bibliothèque-de-projets" et la classe "Projet" est décrite par le biais de l'attribut "est-composé-de" de la classe "Bibliothèque-de-projets" qui prend comme valeur une liste d'instances de la classe "Projet".

3.2.1. La couche objet de MIPS

La couche objet de MIPS implante plusieurs notions empruntés aux Langages Orientés Objets (LOO), aux Représentations Centrées Schéma (RCS) et aux Langages d'Acteurs (LA). La notion de classe se retrouve dans celle d'objet abstrait. Ces objets abstraits sont définis par leurs champs (ou slots). Les champs eux-mêmes sont des objets possédant un certain nombre de facettes. Les objets abstraits peuvent être regroupés en hiérarchies. Ce regroupement est libre dans la mesure où il est défini après la création des objets qui y interviennent. Un même objet peut appartenir à plusieurs hiérarchies. Ceci fait que l'héritage de structure n'est pas supporté afin d'éviter les conflits d'héritage. Lors de l'envoi de messages, la hiérarchie à partir de laquelle on considère l'objet est spécifiée. L'instanciation d'un objet abstrait produit un objet concret. L'instanciation est multiple et profonde.

3.2.2. Les fonctionnalités

Les primitives graphiques permettent d'associer des comportements standards, en termes de réactivité graphique, au divers objets de l'environnement MIPS. Ces primitives graphiques sont écrites en Aïda (un langage d'image portable qui manipule les primitives du bitmap virtuel du système Le-Lisp) et sont activés par l'envoi de messages ("ask", "help", "edit"...).

Les systèmes experts sont définies, crées et gérés en utilisant la couche objet de MIPS. Il existe des structures de systèmes experts définies comme des objets abstraits et regroupés dans une hiérarchie (la typologie des systèmes experts). Cette hiérarchie contiendra à terme différents modèles de système expert selon leur usage : simulation, conception, formation... Le raisonnement du moteur d'un système s'appuie sur les règles (qui sont aussi des objets MIPS) et les objets présents dans le champ "world" du système. Les fonctions de modélisation des raisonnements reposent sur de nombreuses structures de données entièrement réflexives c'est-à-dire des objets systèmes (tels

que les arbres de tâches, les bases de règles, les états...) qui sont modélisés sous la forme de structures de données de même nature que les objets utilisateurs.

Le module Hypertexte défini un ensemble d'outils logiciels permettant un accès intégré à l'information. L'approche est de répondre aux préoccupations des concepteurs qui doivent identifier l'information technico-réglementaire impliquée par un projet, puis être en mesure d'accéder aux informations contenues dans la base précédemment consultée.

Le module de CAO permet une saisie du modèle profond (caractéristiques intrinsèques et extrinsèques des objets). La visualisation devient alors un effet de bord sur la structure de connaissance. Une modification de cette structure (par des règles expertes par exemple) entraîne une modification de la visualisation par l'application de démons.

Le module de visualisation 3D permet de représenter et de manipuler des formes tridimensionnelles. La représentation des objets se fait soit en fil-de-fer soit en rendu de type faces cachées avec source d'éclairage ponctuelle.

3.2. DESCRIPTIF DU MODELE DE DONNEES DE L'ESI

Dans ce paragraphe les classes du modèle de l'ESI générique sont décrites. Pour chacune des classes, le descriptif comporte :

° un contexte qui explicite l'usage de la classe au sein de l'ESI et justifie les choix effectués ;

° la liste des attributs et des méthodes de la classe (données intrinsèques). Pour chacun des attributs, le type et les valeurs possibles sont décrites ;

° les liens entre les classes (données extrinsèques). Les liens entre les classes sont de deux natures :

- les liens de type "généralisation/spécification"' auxquelles sont associées les propriétés d'héritage ;

- les relations qui décrivent le partage de données entre les classes. Etant donné que dans le système MIPS, qui nous sert de plate-forme logicielle d'implémentation et de validation du modèle de l'ESI, les relations entre les classes ne possèdent pas pour l'instant de méthode d'implémentation propre, ces relations sont décrites par l'affectation à un attribut d'une classe une instance (ou liste d'instances) de la classe à laquelle elle est reliée. Ceci se traduit dans la structure par un attribut qui reflète la relation. Les attributs servant à implanter les relations sont référencés par des verbes afin de les distinguer des autres attributs.

Les classes ne correspondent pas nécessairement à des classes d'objets concrets manipulés a sein de l'ESI : elles peuvent aussi correspondre à des abstractions (classes abstraites) permettant de grouper au sein d'une classe les caractéristiques communes aux sous-classes (par exemple, la classe "Présentation" - cf. §3.2.1- regroupe le caractéristiques communs aux objets visibles à l'écran). Les classes abstraites ne sont pas instanciables.

Les classes décrites ci-après, sauf mention contraires, sont des classes qui peuvent être instanciées. A chacune de ces classes est donc rattachée un méthode d'instanciation.

3.2.1. Classe "Présentation"

Elle gère la présentation graphique de tous les objets de l'ESI qui sont visibles à l'écran. Chacun de ces objets (modèles, macro-modèles, bibliothèques de modèles...) possède :

° une partie sémantique qui décrit sa structure propre ;

° une partie syntaxique qui décrit l'aspect, et éventuellement le comportement, qu'il possède à l'écran.

C'est cette partie syntaxique des objets visibles à l'écran que la classe "présentation" modélise. La séparation entre la syntaxe et la sémantique des objets est basée sur le modèle MVC (Model, Vue, Controller) décrit au chapitre I (cf. § I.3.1) : la classe "Présentation" est l'équivalent de la classe "Vue" du modèle MVC.

Contrairement aux outils de CAO, le but dans un ESI n'est pas de donner une représentation à l'échelle d'un objet réel mais une représentation symbolique des entités manipulées. Les utilisateurs, percevant à l'écran les représentations symboliques, ont recours à des clés associatives pour retrouver la sémantique des entités représentées.

La représentation graphique la plus adaptée à la représentation symbolique est l'icône : les icônes peuvent transmettre un flux d'informations important tout en ayant un encombrement réduit (ceci est intéressant pour réduire les contraintes physiques telle que la taille de l'écran).

La classe "Présentation" est une classe abstraite. Elle est décrite comme la classe mère des objets visibles à l'écran afin d'y regrouper les attributs communs de ces objets (cf. Fig. III.4). Ces attributs sont les suivants :

° nom : prend comme valeur une chaîne de caractères qui décrit le nom donné par l'utilisateur aux instances des sous-classes de "Présentation" ;

° coordonnées : contient un doublet représentant les coordonnées à l'écran de l'objet. Ces coordonnées sont exprimées dans le repère de la fenêtre où l'objet apparaît.

° image : contient le nom du fichier qui sert à stocker l'image de l'objet dans un format de points (bitmap).

° sélectionné : permet de stocker et de connaître à tout instant l'état courant de l'objet. Il est de type "booléen" : il prend comme valeurs "vrai" ou "faux".

Les méthodes définies au niveau de la classe "Présentation" sont les méthodes nécessaires à la gestion graphique : déplacer, supprimer, dupliquer...

Par ailleurs, des méthodes qui utilisent les attributs de "Présentation" (hérités par les sous-classes) peuvent être définies au niveau des sous classes afin d'implémenter la relation entre la structure interne et la présentation d'un objet : il s'agit d'implémenter, au niveau des sous-classes, des méthodes qui sont déclenchées à la suite d'une modification de la structure d'un objet et qui ont pour effet une modification de l'aspect externe de l'objet.

Par exemple, une telle méthode est implémentée au niveau des modèles afin de remplacer l'icône du modèle par une icône marquée d'un verrou dès que la valeur "vrai" est affectée à l'attribut "état" du modèle (cf. §3.2.11). Ces méthodes sont importantes dans un ESI puisqu'elles permettent d'améliorer la lisibilité du système étudié en répercutant à l'écran les conséquences des modifications de ses structures internes à l'écran.

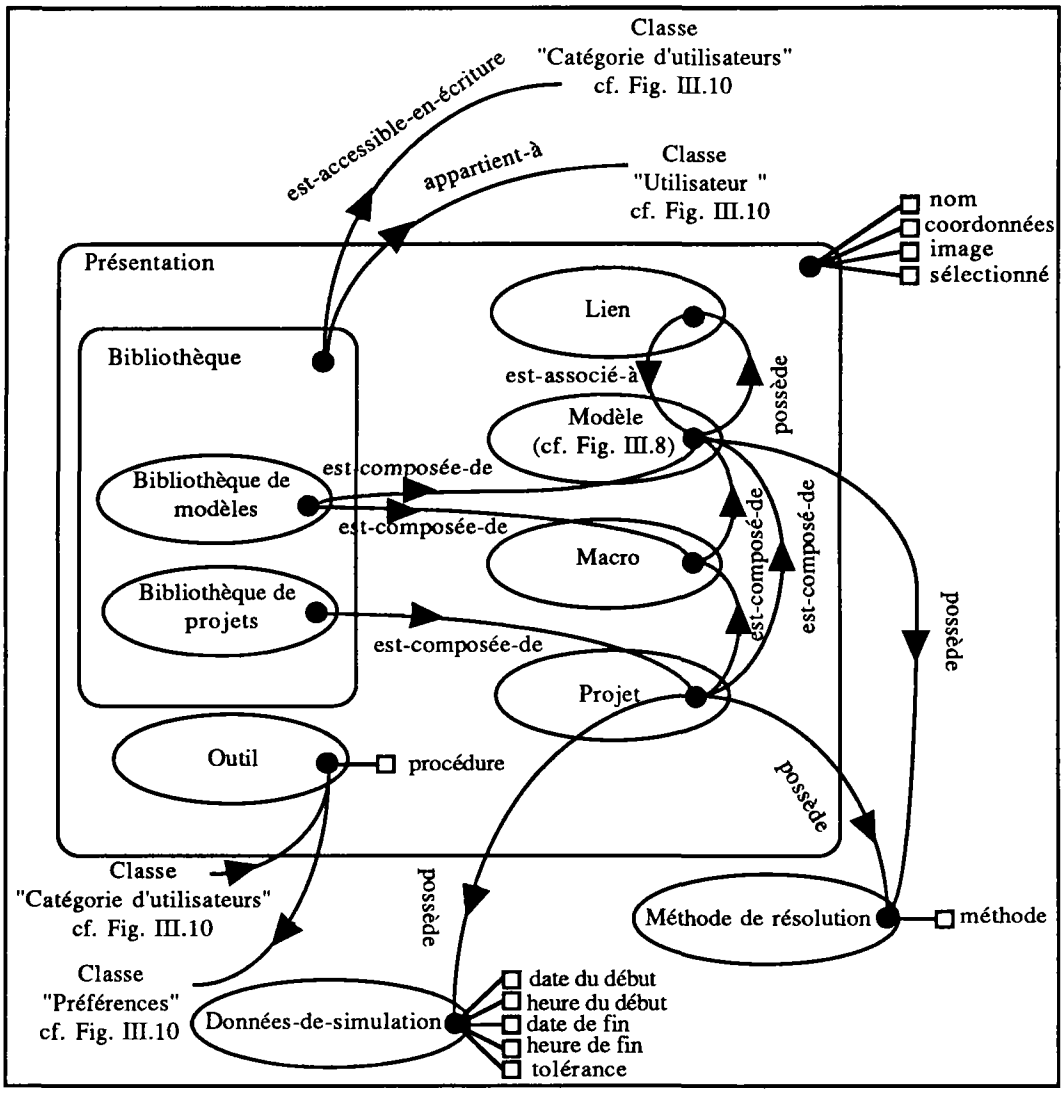


Figure III.4. Structure des sous-classes de "Présentation", et des classes "Méthode de résolution" et "Données de simulation".

3.2.2. Classe "Bibliothèque"

Les bibliothèques sont les structures d'accueil et de rangement des modèles, macro-modèles et projets. Elles doivent être visibles à l'écran afin que les utilisateurs puissent les manipuler. Au sein d'un ESI, différents types d'utilisateurs peuvent avoir accès aux mêmes bibliothèques sans pour autant avoir les même droits : les Up, Ui et Us ont accès en lecture à toutes les bibliothèques (pour consulter et utiliser les composants des bibliothèques), l'accès en écriture (pour supprimer, insérer ou modifier un composant dans une bibliothèque) est défini explicitement par le propriétaire de la bibliothèque.

Deux types de bibliothèques sont prévues au sein de l'ESI :

° les bibliothèque des modèles et des macro-modèles (cf.3.2.3) ;

° les bibliothèques de projets (cf.3.2.4).

Bien que ces deux types de bibliothèques possèdent des comportements identiques, elles accueillent des entités de nature différente : les projets sont des entités qui contiennent toute l'information nécessaire pour lancer des simulations tandis que les modèles et les macro-modèles sont des briques élémentaires (dans le cas des modèles) ou non (dans le cas des macro-modèles) qui peuvent entrer dans la construction de différents projets.

La classe "Bibliothèque" est définie comme une sous-classe de "Présentation", elle hérite ainsi des attributs de "Présentation" (nécessaires pour sa gestion à l'écran). Elle est la classe mère des classes "Bibliothèque-modèles" et "Bibliothèque-projets".

La classe "Bibliothèque" est une classe "abstraite" : la création d'instances de la classe "Bibliothèque" est impossible ; la création d'instances se fait au niveau des sous-classes "Bibliothèque-modèles" et "Bibliothèque-projets".

Les attributs spécifiques de classe "Bibliothèque" sont les suivants :

° appartient-à : prend comme valeur une instance de la classe "Utilisateur", instance qui représente le propriétaire de la bibliothèque. Cet attribut est utilisé, par exemple, afin de vérifier la suppression de la bibliothèque (avec tous ses composants) : seul le propriétaire de la bibliothèque possède ce droit ;

° est-accessible-en-écriture-à : cet attribut est typé et multivalué. Il peut prendre comme valeur une liste d'instances de la classe "Catégorie d'utilisateurs". Cette liste répertorie les catégories d'utilisateurs qui peuvent apporter des modifications à la bibliothèque. Ces modifications peuvent concerner la création, l'insertion (par la technique de copier/coller) ou la suppression de composants. La catégorie d'utilisateur à laquelle appartient le propriétaire de la bibliothèque a, par défaut, accès en écriture à la bibliothèque.

Les méthodes définies au niveau de la classe "Bibliothèque" sont :

° dupliquer : pour créer une copie "profonde" de la bibliothèque. Il s'agit d'une copie qui possède comme composants des copies de tous les composants de la bibliothèque initiale. Cette méthode crée une nouvelle instance de la classe concernée ("Bibliothèque de modèles" ou "Bibliothèque de projets") et déclenche une procédure de saisie des valeurs des attributs de cette instance ;

° supprimer : permet de supprimer la bibliothèque ainsi que ses composants.

3.2.3. Classe "Bibliothèque-modèles"

Les bibliothèques de modèles peuvent contenir :

° des modèles élémentaires (cf. § 3.2.11) ;

° des macro-modèles composés eux même de macro-modèles (cf. § 3.2.5) et/ou de modèles élémentaires.

La possibilité de ranger des entités différentes telles que des modèles élémentaires et des macro-modèles dans une même bibliothèque est justifiée par l'usage de ces entités : bien que ayant une composition différente, elles répondent à un même objectif qui est de reproduire le comportement d'un composant ou d'un système face à certaines sollicitations.

La classe "Bibliothèque-modèles", sous-classe de "Bibliothèque", possède un attribut spécifique unique :

° est-composé-de : attribut typé et multivalué, prend comme valeur une liste d'instances des classes "Modèles" et "Macro".

3.2.4. Classe "Bibliothèque-projets"

Les bibliothèques de projets ont pour but de permettre à l'utilisateur d'archiver ses projets de simulation.

La classe "Bibliothèque-modèles", sous-classe de "Bibliothèque" possède un attribut spécifique unique :

° est-composé-de : attribut typé et multivalué, prend comme valeur une liste d'instances de la classe "Projet".

3.2.5. Classe "Macro"

Les macro-modèles sont des assemblages de modèles élémentaires et/ou de macro-modèles stockés en bibliothèque. Les macro-modèles sont réalisés par l'Up ou l'Ui.

La notion de macro-modèle est liée à l'approche de décomposition récurrente d'un système complexe en éléments plus simples afin de contourner les difficultés de résolution (notamment numériques) : le système est décomposé en éléments dont la modélisation ne pose pas de problèmes particuliers, les modèles des composants du système sont alors regroupés dans un macro-modèle censé reproduire le comportement du système global (en réalité, le macro-modèle risque de ne pas être fidèle au système global si les modèles qui le composent ne prennent pas en compte leurs interactions mutuelles).

Un des avantages de l'utilisation de la notion de macro-modèle est la séparation entre le fonctionnement interne des modèles qui le composent et le couplage entre ces modèles : un macro-modèle est décrit par ses composants et par le couplage entre les composants définie à travers une interface de communication. Par conséquent, un élément du macro-modèle peut être modifié sans affecter le fonctionnement global du macro-modèle (à condition que l'interface de communication avec les autres modèles soit maintenue). Ceci signifie que le macro-modèle peut être amélioré graduellement (par

remplacement des modèles qui le composent par des modèles plus élaborés) tout en minimisant l'effort de restructuration qui en découle.

Une des difficultés de l'utilisation de macro-modèle est le maintien de sa cohérence globale : la cohérence individuelle des composants du macro-modèle est nécessaire pour assurer la cohérence du macro-modèle mais elle n'est pas suffisante . Par exemple, les hypothèses de deux composants du macro-modèle peuvent être incompatibles. Au sein de l'ESI, le maintien de la cohérence dans un assemblage (et, en particulier, dans un macro-modèle) s'effectuent à l'aide de règles de cohérence qui utilisent, dans ce cas particulier, les attributs de la classe "Hypothèse" (cf. 3.2.14).

La classe "Macro" est une sous-classe de "Présentation" puisqu'elle représente des entités visibles à l'écran. Ses attributs spécifiques sont :

° est-composé-de : attribut typé et multivalué, prend comme valeur une liste d'instances des classes "Modèle" et "Macro".

La méthode définie au niveau de "Macro" est :

° éclater : cette méthode permet de remplacer, au cours d'une session de travail, la structure et la présentation d'un macro-modèle par les structures et les présentation de ses composants.

3.2.6. Classe "Projet"

Un projet est un assemblage de modèles élémentaires et/ou de macro-modèles associé à des données de simulation : il est réalisé par l'Uf à partir du travail de modélisation réalisé en amont par l'Up et l'Ui. Un projet est l'entité qui sert à l'Uf pour effectuer des études paramétriques.

Un projet représente un système d'équations algébro-différentielles (S) composé des systèmes d'équations algébro-différentielles (Si) associés à chacun des modèles élémentaires qui le composent. Par ailleurs, les liens entre les modèles composant un projet se traduisent par des couplages entre les systèmes d'équations (Si).

Bien qu'à chaque modèle élémentaire soit associée une ou plusieurs méthode de résolution numérique adaptée, ceci n'est pas suffisant pour en déduire une méthode de résolution adaptée au projet : même au cas où il existe une méthode de résolution commune à tous les modèles composant le projet, les couplages entre les modèles mais aussi les données de simulation associées au projet peuvent faire en sorte que cette méthode ne soit pas adaptée à la résolution du projet dans sa globalité. L'auteur du projet doit par conséquent associer une méthode de résolution numérique adaptée au projet.

La classe "Projet" est une sous-classe de "Présentation" puisqu'elle représente des entités manipulables à l'écran ; elle possède des attributs spécifiques :

° est-composé-de : attribut typé et multivalué, prend comme valeur une liste d'instances des classes "Modèle" et "Macro" :

° possède-des-données-de-simulation : attribut typé et monovalué, prend comme valeur une instance de la classe "Données-de-simulation" :

° possède-une-méthode-de-résolution : attribut typé et multivalué, prend comme valeurs une liste d'instances de la classe "Méthode de résolution" (cf. §3.2.16) ; décrit les méthodes de résolution numérique du projet.

A la classe "Projet" est rattachée une méthode unique :

° simuler : permet d'exécuter la simulation en utilisant l'assemblage décrit pour le projet ainsi que les données de simulation. Dans le cas de l'utilisation de code de simulation existant, cette méthode génère un fichier d'entrée pour ce code à partir de la structure du projet.

3.2.7. Classe "Données-de-simulation"

Les données de simulation représentent les données globales nécessaires pour compléter l'information relative à un projet afin de pouvoir exécuter des simulations. Les données de simulation n'incluent pas les valeurs attribuées aux paramètres des modèles.

Les données de la simulation définies au niveau de l'ESI générique (donc indépendantes des codes de simulation) sont d'une part celles qui fixent le début et la fin de la simulation et, d'autre part, celles qui définissent le seuil de convergence des modèles (tolérance). D'autres données de simulation relatives à des codes de simulation spécifiques sont souvent nécessaires. Elles seront définies lors de l'adaptation de l'ESI générique à ces codes (par ajout de sous-classes à la classe "Données-de-simulation").

Les attributs relatifs aux dates de début et de fin de simulation sont utilisés pour l'accès à des fichiers de données externes (par exemple, fichiers météorologiques).
Les attributs relatifs aux temps du début et de fin de la simulation sont nécessaires pour la synchronisation de l'horloge utilisée dans le modèle de temps (cf. §3.2.17).

Afin d'éviter toute interprétation erronée de la signification des chiffres désignant les dates et les heures, les formats utilisés sont des formats normalisés : la norme suivie est la norme ISO 8601 spécifiant la représentation numérique de l'information liée à la date et à l'heure du jour (cf. Annexe 1).

Les attributs de la classe "Données-de-simulation" sont :

° date-du-début : prend une valeur décrivant la date du début de la simulation dans le format des dates ;

° heure-du-début : prend une valeur décrivant l'heure du début de la simulation dans le format des heures ;

° date-de-fin : prend une valeur décrivant la date de fin de la simulation dans le format des dates ;

° heure-de-fin : prend une valeur décrivant l'heure de fin de la simulation dans le format des heures.

° tolérance : prend comme valeur un réel définissant la tolérance relative z associée au projet. Il s'agit de la valeur qui définit le seuil de convergence pour les variables des modèles. La convergence du modèle est supposée atteinte quand, pour toutes ses variables V, on peut écrire $V_i - V_{i-1} > z$ (i étant le nombre d'itérations pour un pas de temps donné).

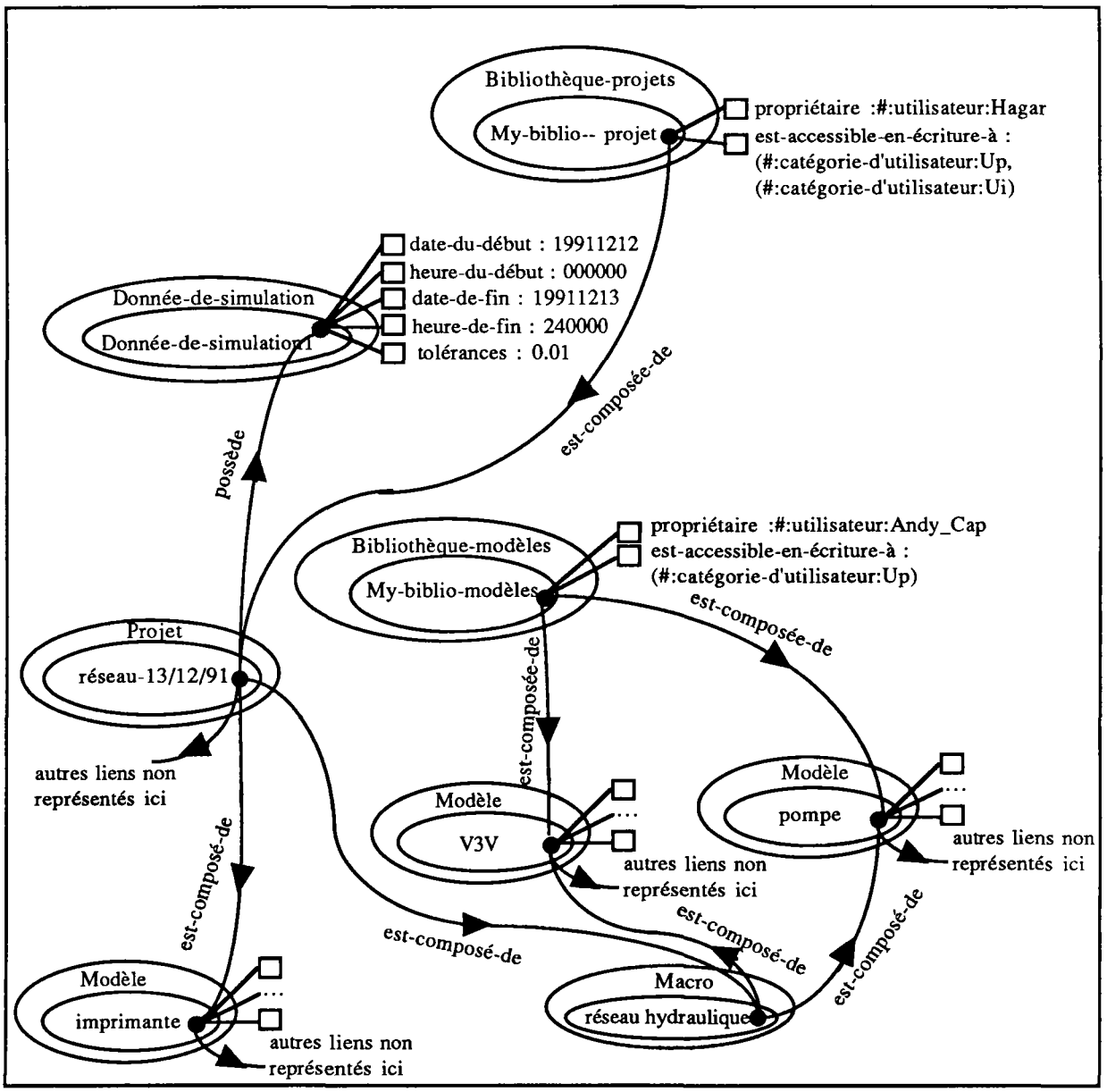


Figure III.5. Exemple d'instanciation des classes "Bibliothèque-projets", "Bibliothèque-modèles", "Modèles", "Macro", "Projet" et "Données-de-simulation".

3.2.8. Classe "Grandeur"

Les *grandeurs* physiques sont des entités de base qui entrent dans la composition des modèles : c'est en décrivant les relations entre des grandeurs physiques (relations établies par l'expérience ou par l'analyse) que des théories, d'application plus ou moins large, ont été construites afin de représenter au mieux le réel.

Les grandeurs pouvant se comparer mutuellement forment des ensembles (par exemple, les longueurs, les distances, les hauteurs, les longueurs d'onde...).

Pour des raisons en partie arbitraires, certaines grandeurs sont choisies comme grandeurs de base, considérées conventionnellement comme indépendantes. Les autres grandeurs, appelées grandeurs dérivées, se déduisent des grandeurs de base par des relations entre grandeurs. L'ensemble des grandeurs de base, des grandeurs dérivées et des relations qui les lient forme un système de grandeurs. L'ensemble de grandeurs le plus utilisé est celui basé sur les sept grandeurs de base : longueur, masse, temps, courant électrique, température, quantité de matière et intensité lumineuse.

Les grandeurs sont représentées par des symboles constitués généralement par une seule lettre de l'alphabet latin ou grec avec éventuellement des indices (par exception, certains symboles composés de plusieurs lettres sont parfois employés). Une grandeur peut être représentée par des symboles différents(par exemple, le moment d'inertie est représenté par I ou J).

La dimension d'une grandeur Q est l'expression $A^a B^b C^g...$ où A, B, C... indiquent les dimensions des grandeurs de base A, B, C... et où a, b, g... sont appelés les exposants dimensionnels de la grandeur Q. Une grandeur dont tous les exposants dimensionnels sont nuls est dite grandeur sans dimension. Dans le système fondé sur les sept grandeurs de base décrit ci-avant, les dimensions de base peuvent être indiquées respectivement par L, M, T, I, Q, N ET J et la dimension d'une grandeur Q devient :

$$\dim Q = L^a M^b T^g I^d T^e N^x J^h.$$

Par exemple :	Grandeur	Dimension
	vitesse	LT^{-1}
	force	LMT^{-2}
	énergie potentielle	L^2MT^{-2}
	énergie cinétique	L^2MT^{-2}
	densité relative	sans dimension

Les attributs de la classe "Grandeur", défini au sein de l'ESI générique, reproduisent les principales caractéristiques d'une grandeur physique. Ces attributs sont les suivants :

° nom : prend comme valeur une chaîne de caractères représentant le nom de la grandeur.
Par exemple : force ;

° symbole : prend comme valeur une chaîne de caractères (généralement, une seule lettre de l'alphabet latin ou grec) représentant le symbole de la grandeur.
Par exemple : F ;

° définition : définit, en texte libre, la grandeur.
Par exemple : la force résultante agissant sur un corps est égale à la dérivée de la quantité de mouvement par rapport au temps ;

° possède-unité-SI : prend comme valeur une instance de la classe "Unité" et représente l'unité de la grandeur dans le système SI (cf. § 3.2.9).
Par exemple : #:unité:newton

° possède-autre-unités : attribut typé et multivalué qui prend comme valeur une liste d'instances de la classe "Unité" et représente les unités (autre que l'unité dans le système SI) qui peuvent être utilisées pour la grandeur en raison de leur importance pratique.

Par exemple : pour #:grandeur:masse, la valeur de l'attribut autre-unité peut être (#:unité:tonne).

La classe "Grandeur" est utilisée par la classe "Dictionnaire-de-grandeurs" (cf. §3.2.10).

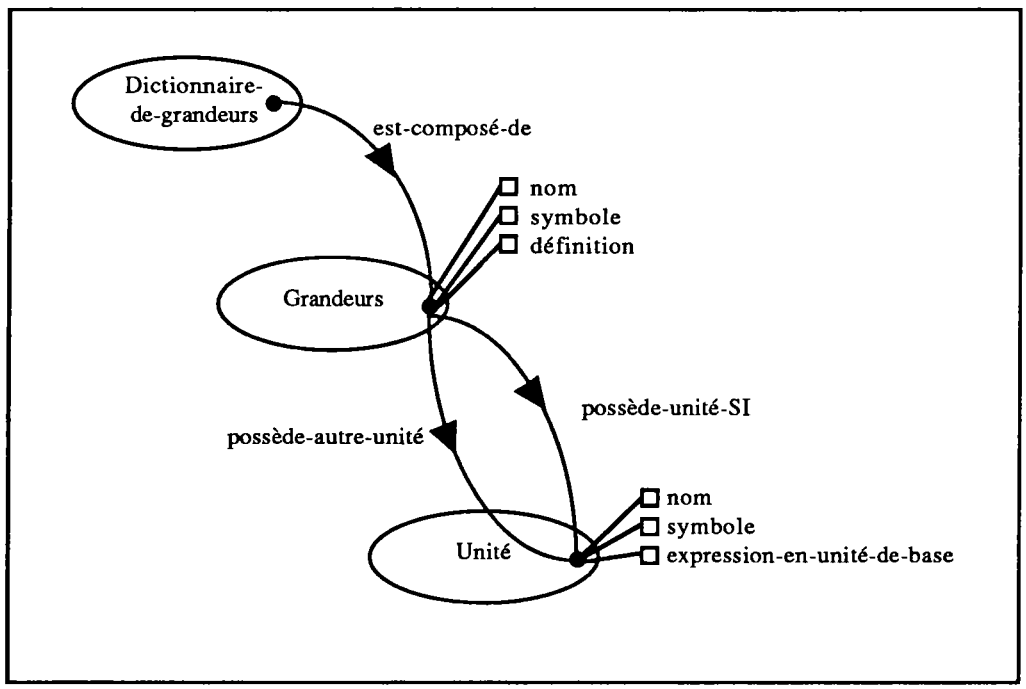


Figure III.6. Structure des classes "Dictionnaire-de-grandeurs", "Grandeur" et "Unité".

3.2.9. Classe "Unité"

Dans un ensemble de grandeurs pouvant se comparer mutuellement, une valeur particulière d'une grandeur particulière est choisie arbitrairement comme grandeur de référence appelée l'*unité* : toute grandeur de cet ensemble peut alors être exprimée, en fonction de cette unité, par le produit de cette unité par un nombre.

Pour un système de grandeur donné, un *système d'unités* peut être établi : un système d'unités comprend alors un ensemble d'unités de base (relatives aux grandeurs de base) et d'unités dérivées déterminées par leurs équations de définition et les facteurs de proportionnalité (par exemple 1 Wh =3,6*10^3 J).

Un système d'unités est dit cohérent par rapport à un système de grandeurs quand les équations entre valeurs numériques ont la même forme que les équations correspondantes entre grandeurs. Le système SI est cohérent par rapport au système de grandeurs fondé sur les sept grandeurs de base (cf. norme NF X 02-006). Les unités de base correspondantes aux grandeurs de base sont :

Grandeur	Nom de l'unité de base	Symbole de l'unité de base
longueur	mètre	m
masse	kilogramme	kg
temps	seconde	s
intensité du courant électrique	ampère	A
température thermodynamique	kelvin	K
quantité de matière	mole	mol
intensité lumineuse	candela	cd

Pour éviter des valeurs numériques faibles ou élevées, on peut ajouter des multiples et sous-multiples décimaux aux unités : ils sont formés par ajout de préfixes (par exemple : 10^9 = giga ; 10^2 = hecto...).

Les attributs de la classe "Unité" sont :

° nom : prend comme valeur une chaîne de caractères représentant le nom de l'unité. Par exemple : newton ;

° symbole : prend comme valeur une chaîne de caractères représentant le symbole de l'unité.
Par exemple : N ;

° expression-en-unité-de-base : exprime l'unité en fonction des sept unités de base. Cette expression peut comporter des facteurs multiplicatifs mais aussi des facteurs additifs.
Par exemple : 1 N = 1 $m.kg.s^{-2}$.

3.2.10. Classe "Dictionnaire-de-grandeurs"

Vu l'importance des grandeurs physiques dans tout travail de modélisation, un recueil cohérent des grandeurs les plus utilisées associées à leurs unités nous parait nécessaire dans l'ESI : il permet d'offrir à l'utilisateur une assistance d'une part lors de la création d'un nouveau modèle (choix d'unités cohérentes, vérification de la formulation du modèle par rapport aux dimensions des grandeurs...) et, d'autre part, lors de l'utilisation des modèles existants (vérification de la validité des connexions entre les variables...).

Le recueil des grandeurs est structuré sous forme de dictionnaire et il est basé sur un travail de normalisation déjà réalisé dans les normes NF X 02-201 à 213 : ces normes décrivent les grandeurs les plus utilisées dans les domaines suivants : espace et temps, phénomènes périodiques et connexes, mécanique, thermique, électricité et magnétisme, rayonnements électromagnétiques et optique, acoustique, chimie physique et physique moléculaire, physique atomique et nucléaire, réactions nucléaires et rayonnements ionisants, physique de l'état solide.

Le dictionnaire des grandeurs contient une liste prédéfinies de grandeurs dans les domaines cités ci-avant. Cette liste peut être enrichie par les utilisateurs de l'ESI tout en gardant une cohérence globale : il s'agit de vérifier si la nouvelle grandeur n'est pas redondante avec une grandeur existante (la vérification se fait relativement à la dimension de la grandeur ; la liste des grandeurs existantes partageant la même dimension avec la grandeur introduite est proposée à l'utilisateur).

La classe "Dictionnaire-de-grandeurs" possède un attribut unique :

° est-composé-de : cet attribut est multivalué et prend comme valeur une liste d'instances de la classe "Grandeur" (cf. Fig. III.7).

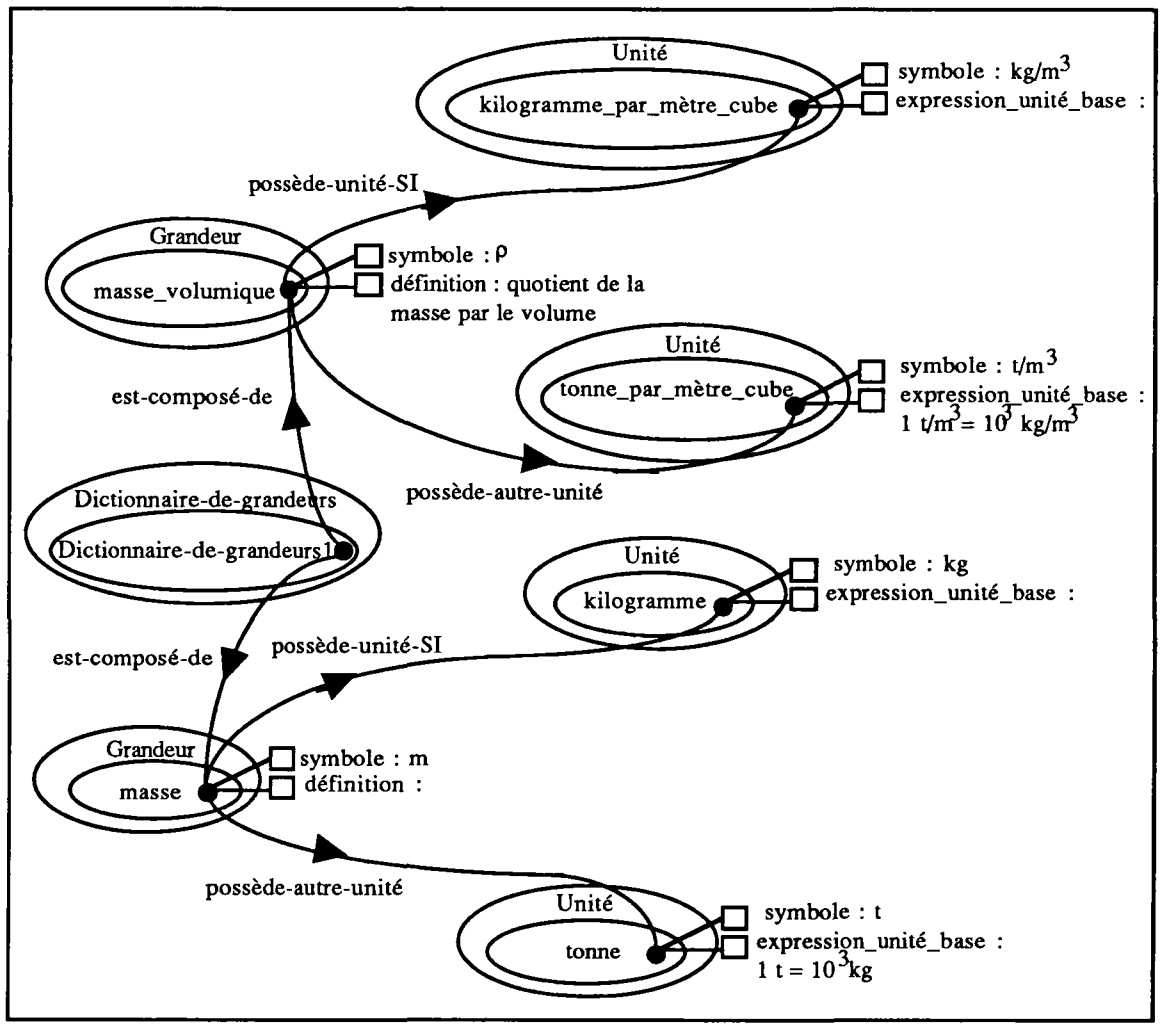


Figure III.7. Exemple d'instanciation des classes "Dictionnaire-de-grandeurs", "Grandeur" et "Unité".

Les méthodes rattachées à la classe "Dictionnaire-de-grandeurs" sont insertion et suppression. Ces méthodes permettent respectivement d'insérer et de supprimer une grandeur du dictionnaire. La méthode insertion déclenche un test pour vérifier si la dimension de la grandeur insérée est déjà utilisée. Si c'est le cas, la liste des grandeurs utilisant cette dimension est proposée à l'utilisateur avant de lui demander de confirmer l'insertion. Ceci permet d'éviter les redondances dans le dictionnaire.

3.2.11. Classe "Modèle"

Un modèle, tel qu'il a été défini ci-avant (cf. §2.6 du Chapitre II) est la combinaison d'une part d'un système d'équations algébro-différentielles associé à un certain nombre de valeurs initiales et/ou de sollicitations et, d'autre part, d'hypothèses sous-jacentes. Une méthode de résolution numérique adaptée peut être associée au modèle.

Le modèle reproduit le comportement d'un ou plusieurs objets associés à un ou plusieurs phénomènes s'y déroulant.

Un modèle peut être validé par des méthodes différentes : les trois principales méthodes de validation qui concernent les modèles tels qu'elles ont été définis (cf. § 3.6 du Chapitre II) sont :

° validation par rapport à des modèles déjà validés ;

° validation par simplification du modèle (choix de valeurs de variables particulières rendant la résolution analytique possible) et comparaison entre les sorties du modèle et la solution analytique ;

° validation expérimentale.

La structure de la classe "Modèle" doit permettre d'exprimer les informations relatives à un modèle, et par conséquent le capital de connaissance que représente le modèle, sans difficulté. La création par l'Up d'un modèle dans l'ESI revient à créer une instance de la classe "Modèle". Cette instance est ensuite utilisée par des instances des classes "Bibliothèque-modèles" (cf. §3.2.3), "Macro" (cf. §3.2.5), "Projet" (cf. §3.2.6)...

La classe "Modèle" est définie comme une sous-classe de la classe "Présentation" afin qu'elle puisse hériter des attributs de cette dernières.

Les attributs de la classe "Modèle" sont les suivants :

° nom : cet attribut prend comme valeur une chaîne de caractères qui représente le nom donné au modèle par son auteur ;

° auteur : prend comme valeur une instance de la classe "Utilisateur", instance qui représente l'auteur du modèle ;

° résumé : prend comme valeur une description en texte libre de l'objectif du modèle, de son contexte d'utilisation...

° échelle : définit qualitativement l'échelle d'analyse choisie pour le modèle. La valeur de cet attribut monovalué appartient à l'ensemble {détaillé, simple, simplifié}. Pour le moment, il n'existe pas dans le domaine de la modélisation une normalisation des notions de modèle détaillé ou de modèle simplifié. Ces notions portent toujours une grande part de subjectivité. A défaut d'avoir des définitions claires de ces notions (qui peuvent alors déboucher sur une automatisation de l'évaluation de l'échelle du modèle en fonction de ses équations et ses hypothèses), l'estimation de l'échelle est laissée à la charge de l'auteur du modèle ;

° méthode de validation : attribut multivalué et typé qui prend ses valeurs dans la liste {comparative, par-simplification, expérimentale} ;

° possède-modèle-amont : attribut multivalué et typé qui prend comme valeur une liste d'instances de la classe "Modèle" qui représente les modèles nécessaires pour l'utilisation du modèle étudié ;

° possède-lien : attribut multivalué et typé qui prend comme valeur une liste d'instances des classes "Lien-orienté" ou "Lien-non-orienté" (cf. §3.2.15) ;

° état : attribut monovalué et typé qui prend comme valeur "vérouillé" ou "accessible" et permet de décrire si l'auteur du modèle interdit ou non sa modification et sa suppression par d'autres utilisateurs ;

° modélise-composant : attribut multivalué et typé. Prend comme valeur une liste d'instances de la classe "Composant technologique" ;

° modélise-phénomène : attribut multivalué et typé. Prend comme valeur une liste d'instances de la classe "Phénomène" ;

° possède-méthode-de-résolution : attribut multivalué et typé. Prend comme valeur une liste d'instances de la classe "Méthode-de-résolution" ;

° utilise-variables : liste d'instances de la classe "Variable" qui décrit les variables utilisées dans le modèle ;

° possède-formulation : une instance de la classe "Formulation" qui décrit la formulation du modèle ;

° est-associé-à-hypothèses : liste d'instances de la classe "Hypothèse" qui décrit les hypothèses du modèle. A cet attribut est rattaché un démon qui teste si la liste des hypothèses ne comprend pas des hypothèses contradictoires (cf. § 3.2.14).

Les méthodes définies au niveau de la classe "Modèle" sont :

° création : permet à l'Up de créer des modèles. Cette méthode entraîne la création des éléments composant le modèle (formulation, variable, hypothèse) et ceci par instanciation des classes correspondantes. La création d'un modèle est suivie par son insertion dans une bibliothèque de modèles (cf. § 3.2.3) ;

° suppression : permet de supprimer le modèle de la bibliothèque. Cette méthode ne peut être déclenchée que par un utilisateur ayant un droit d'écriture dans la bibliothèque ;

° copier/coller : permet de créer une copie du modèle puis de rattacher cette copie à une bibliothèque de modèles qui peut être différente de la bibliothèque d'origine.

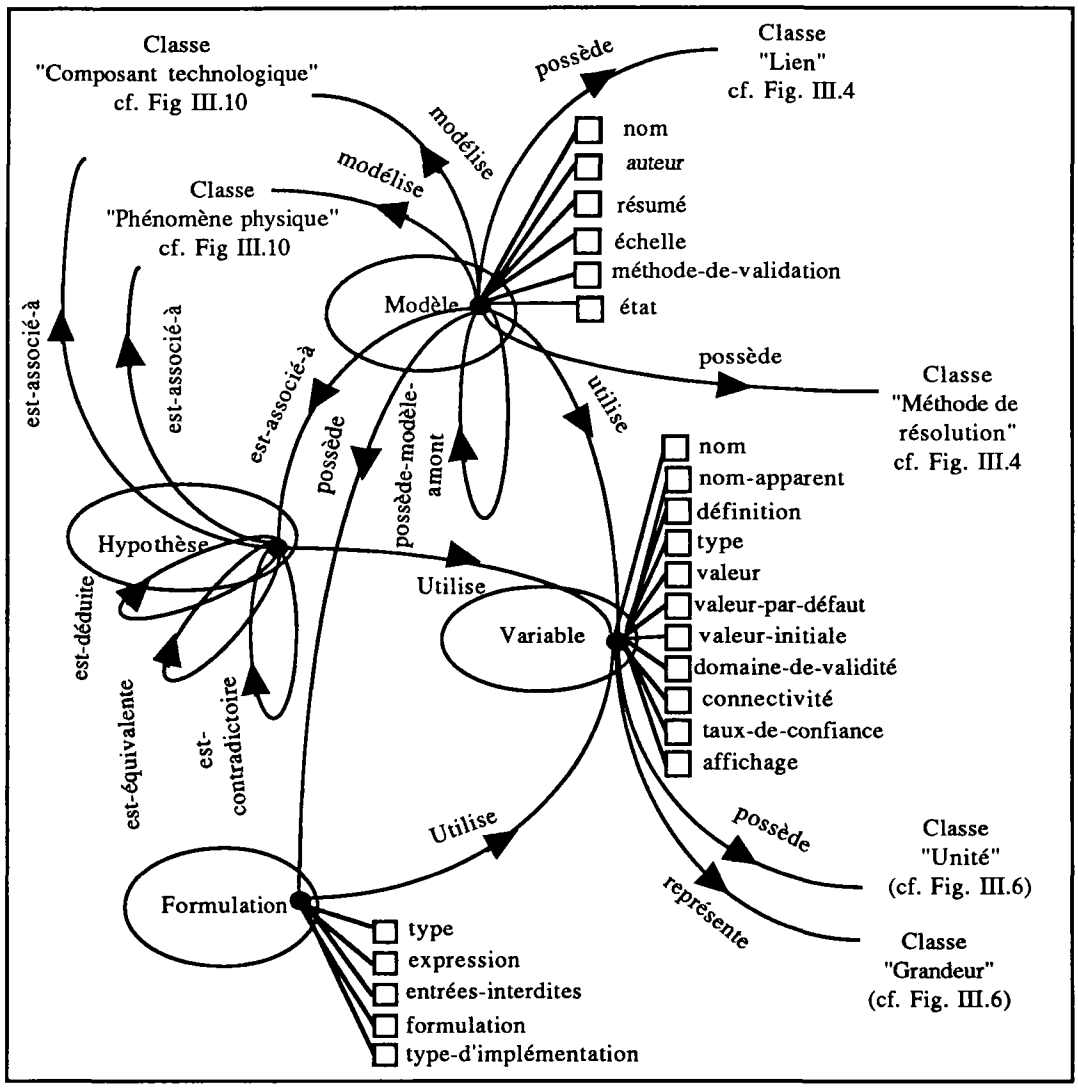


Figure III.8. Description des classes "Modèle", "Hypothèse", "Variable" et "Formulation".

3.2.12. Classe "Variable"

Les variables d'un modèle sont utilisées par sa formulation comme elles peuvent entrer dans l'expression des hypothèses du modèle.
Lors de l'envoi du message "création" à la classe "Modèle", les instances correspondantes de la classe "Variable" sont créées (ainsi que les instances des classes "Formulation" et "Hypothèse"). Certains attributs des instances de la classes "Variable" sont renseignés par l'auteur du modèle et d'autres (comme la valeur des paramètres) le sont par les utilisateurs du modèle.

Les attributs de la classe sont les suivants :

° nom : cet attribut prend comme valeur une chaîne de caractères qui définit la manière avec laquelle la variable sera désignée au sein du modèle ou par le système;

° nom-apparent : cet attribut prend comme valeur une chaîne de caractères qui définit le nom de la variable tel qu'il sera visible lors de l'utilisation du modèle ;

° définition : cet attribut contient un descriptif en texte libre de ce que la variable représente sur le plan physique ;

° type : cet attribut permet de définir le type de la variable afin de distinguer les variables calculées par le modèle (variable de sortie) de celles en entrée du modèle. Les variables en entrée du modèle peuvent être dynamiques (variables d'entrée) ou avoir une valeur constante durant la durée de la simulation (paramètres). Les dérivées du modèle sont les variables pour lesquelles une valeur initiale doit être associée à leur dérivée dans le temps en vue de la résolution numérique du système d'équations. La valeur de l'attribut est spécifiée lors de l'utilisation du modèle. Ceci signifie qu'une variable désignée comme une entrée lors d'une session d'utilisation peut être désignée comme une sortie lors d'une autre session d'utilisation. L'attribut est monovalué et typé : il prend sa valeur dans la liste suivante {paramètre, entrée, sortie, dérivée} ;

° valeur : cet attribut permet à l'utilisateur d'un modèle de définir la valeur de la variable. Cette valeur est statique dans le cas des paramètres et elle est dynamique dans le cas des variables d'entrée et les variables de sortie. L'attribut est monovalué et typé : l'auteur d'un modèle peut définir pour chaque variable utilisée, le type de la valeur admise {entier, réel...}.

  Par ailleurs, la "facette spécifique" de l'attribut valeur est utilisée pour définir le "temps de réaction de la valeur" : le temps de réaction détermine le temps nécessaire à une variable pour changer de valeur après avoir reçue une nouvelle valeur. Ce temps peut être nul ce qui signifie que la valeur changera instantanément comme il peut avoir une durée ce qui signifie que le changement ne sera pas effectif que si la nouvelle valeur persiste pendant cette durée. Cette facette permet de filtrer des incohérences dues à des erreurs de résolution numériques ;

° valeur-par-défaut : une valeur par défaut peut être attribuée par l'auteur du modèle à une variable. Au cas où l'attribut "valeur" de la variable n'est pas renseigné par l'utilisateur du modèle, c'est la valeur par défaut qui est prise en compte par le modèle lors des calculs ;

° valeur-initiale : une valeur initiale doit être attribuée par l'utilisateur du modèle aux variables d'entrée. Pour les équations différentielles la valeur initiale correspond à la valeur en début de simulation, pour les équations algébriques cette valeur correspond à une première estimation de la solution. Pour les variables de sortie et les paramètres, cet attribut n'est pas utile ;

° domaine-de-validité : à chaque variable l'auteur du modèle peut rattacher un domaine de validité. Ce domaine indique l'intervalle dans lequel la variable peut prendre ses valeurs au regard des limites du modèle. Cet attribut est monovalué et typé : il prend comme valeur unique un couple représentant la borne minimale et maximale de l'intervalle ;

° représente-grandeur : cet attribut désigne, si elle existe, la grandeur physique décrite par la variable. Cet attribut est monovalué et typé : il peut prendre comme valeur une instance de la classe "Grandeur" sélectionnée dans le "Dictionnaire de grandeurs" ou rester sans valeur au cas où la variable ne représente pas de grandeur physique (variable interne, variable de calcul...) ;

° possède-unité : cet attribut prend comme valeur une instance de la classe "Unité". Au cas où la variable a été associée à une grandeur physique l'unité, dans le système SI, de cette grandeur est affectée par défaut à la variable (ceci revient à affecter à l'attribut "possède-unité" de la variable la valeur de l'attribut "unité-SI" de la grandeur associée).

Par exemple : si la variable #:variable:rho est associée à la grandeur #:grandeur:masse_volumique, alors l'attribut unité de #:variable:rho prend comme valeur #:unité:kg/m^3.

La valeur par défaut de l'attribut "unité" peut néanmoins être modifiée lors de l'utilisation ce qui revient à affecter à cet attribut une autre instance de la classe "Unité" (et, par conséquent renseigner les attributs "nom-unité", "symbole" et "expression-en-unité-base"). La modification de la valeur par défaut déclenche un démon qui teste la cohérence de la nouvelle unité relativement à la grandeur associée à la variable : il s'agit de vérifier si l'attribut "expression-en-unité-base" de la nouvelle unité est cohérent avec la dimension de la grandeur.

Dans l'exemple précédent, le remplacement de #:unité:kg/m^3 par l'instance de la classe unité ayant comme attributs :

nom-unité : "tonne_par_mètre_carré"
symbole : "t/m^2"
expression-en-unité-base : "1t/m^2 = 10^3 kg/m^2"

est jugé non-cohérent puisque 10^3 kg/m^2 ne correspond pas à la dimension ML^{-3} de #:grandeur:masse_volumique.

Le changement d'unité de la variable entraîne la conversion automatique des attributs "valeur", "valeur initiale", "valeur par défaut" et "domaine de valeur" de la variable dans la nouvelle unité ;

° connectivité : cet attribut définit les variables qui peuvent être connectées à la variable en cours de définition. L'auteur du modèle sélectionne, parmi les variables de modèles existants en bibliothèque et partageant la même dimension avec la variable en cours de définition, celles qu'il juge susceptibles d'être reliées à cette variable.
L'attribut est multivalué et typé : il prend comme valeurs des éléments d'une liste d'instances de la classe "Variable".

Par exemple, pour #:variable:température d'air d'un modèle de "zone", l'attribut "connectivité" peut prendre les valeurs {#:variable:température_ambiante, #:variable:ti} sélectionnées par l'auteur du modèle dans la liste contenant toutes les variables des modèles en bibliothèques ayant comme dimension Q ;

° affichage : attribut monovalué et typé qui prend comme valeur "vrai" ou "faux". Il indique si la variable est visible lors de l'utilisation du modèle. L'utilisation de cet attribut permet de distinguer la complexité interne du modèle de sa complexité lors de l'usage : certaines variables du modèle peuvent être déclarée non visibles ce qui permet de faciliter la manipulation du modèle ;

° taux-de-confiance : cet attribut représente une estimation du taux de confiance qu'on peut accorder à la valeur d'une variable. Il permet de calculer, dans un assemblage donné, le taux de confiance des variables de sortie afin de les comparer à la précision requise au regard du but de simulation. Les taux de confiance sont propagés dans l'assemblage en fonction des liens entre modèles.

Le taux de confiance des variables de sortie d'un modèle est calculé en fonction :

- du taux de confiance associé aux paramètres. Ce taux de confiance représente une estimation donnée par l'utilisateur de la confiance accordée à la source d'information d'où la valeur du paramètre provient (expérimentation, références...) ;

- du taux de confiance associé aux variables d'entrée. Ce taux de confiance est égal aux taux de confiance associé aux variables de sortie du modèle amont ayant des connexions avec les variables d'entrée du modèle étudié.

La méthode de calcul du taux de confiance peut être la méthode de propagation des erreurs : cette méthode consiste à estimer la variance de la grandeur calculée en cumulant les variances de chaque variable pondérées par la sensibilité de la grandeur calculée relativement à la variable considérée (on suppose une distribution normale des erreurs sur les variables - cf. [Für91]).

Pour une fonction $y(x_i)$, où les x_i sont des variables indépendantes, l'écart type S est :

$$S^2(y) = \sum (\partial y/\partial x_i)^2 \, S^2(x_i) \qquad \text{(Eq. III.1)}$$

L'estimation des $\partial y/\partial x_i$ doit se faire numériquement dans l'intervalle étudié.

3.2.13. Classe "Formulation"

La formulation représente la description interne du modèle. Trois types de formulations peuvent être distingués :

° la formulation équationnelle décrit un système d'équations algébro-différentielles non-orienté d'un modèle : ceci permet d'avoir une formulation indépendante des conditions associées. Elle peut être écrite dans une forme implicite :

$f(x,\partial x/\partial t,p) = 0$

où f est un ensemble de n relations
x est un vecteur de n variables
$\partial x/\partial t$ est la dérivée par rapport au temps de x
et p un vecteur de n paramètres.

Le modèle est défini par sa formulation et par ses conditions associées. Une formulation peut être partagée par plusieurs modèles.

Par exemple, la formulation Q-K(t1-t2) =0 peut être utilisée pour calculer

° Q en fonction de t1 et t2 ;
° t1 en fonction de Q et t2 ;
° t2 en fonction de Q et t1.

Un modèle est, par exemple, l'association de cette formulation à la condition : calcul de Q en fonction des entrées t1 et t2.

Dans cet exemple simple, les trois conditions associées sont "bien posées" (permettent de calculer, dans un intervalle donné, une solution unique) ; dans un

cas plus complexe, toutes les combinaisons ne sont pas nécessairement bien posées : ceci est par exemple le cas où l'inverse d'une fonction n'est pas définie dans l'intervalle étudié ;

° la formulation logique permet d'exprimer le modèle sous forme de règles de logiques entre les variables. Par exemple :

si #:variable:température - #:variable:température de consigne > 0
alors #:variable:signal_régulateur = 0.

° la formulation en jeux de données permet de formuler un modèle sous la forme d'un ensemble de points, généralement résultant d'expériences.

Les formulations équationelle et logique décrivent des relations entre des variables (définies comme des instances de la classe "Variable") indépendamment du contexte d'utilisation : ce contexte est défini au niveau des variables (par l'intermédiaire de l'attribut "type" qui prend une des valeurs suivantes : entrée, sortie ou paramètre - cf. § 3.2.12). C'est donc lors du choix des variables de sortie (variables qui sont calculées en fonction des variables d'entrée et des paramètres) que les conditions associées à la formulation sont définies.

Toutes les combinaisons des conditions associées ne sont pas nécessairement pertinentes. L'auteur du modèle peut contraindre une variable à être d'un certain type.

L'implémentation de la formulation des modèles peut être réalisée dans des langages différents : bien que la majorité des modèles existants utilise des langages généraux (Fortran...) ou des langages de simulation (GPSS...), l'implémentation de la formulation d'un modèle peut être réalisée en tant que méthode associée à la classe "Formulation". Ceci permet de tirer profit de la non-redondance du code des environnements orientés objet : les formulations implémentées sous-forme de méthodes d'une classe sont hérités par les sous-classes. L'intégration au sein de l'ESI de ces deux types d'implémentation doit être possible afin d'une part de permettre de traiter les modèles existants qui représentent un acquis important de la communauté scientifique et, d'autre part, de permettre l'utilisation d'environnements innovants.

Les attributs de la classe "Formulation" sont les suivants :

° type : cet attribut désigne le type de la formulation du modèle. Il est monovalué et typé ; il prend une valeur de l'ensemble {équationel, logique, données}. Un démon rattaché à l'attribut permet, au cas ou la valeur "donnée" est sélectionnée, de saisir le nom du fichier contenant les données ;

° utilise-variables : cet attribut prend comme valeur une liste d'instances de la classe "Variable", liste qui regroupe les variables utilisées dans la formulation du modèle. Lors de la création du modèle, ces variables ne sont pas orientées (l'attribut "type" des variables n'est pas renseigné). C'est lors de l'utilisation du modèle que ces variables sont orientées et, par conséquent, que le contexte d'utilisation du modèle est fixé ;

° expression : cet attribut est une chaîne de caractères qui représente l'expression des équations du modèle. Ces équations sont des relations implicites, logiques ou mathématiques (algébriques, différentielles...), entre les variables utilisées. Des variables qui n'ont pas été déclarées (par l'intermédiaire de l'attribut "utilise-variables") ne peuvent pas être utilisées dans l'expression.
Un démon rattaché à l'attribut permet de tester la cohérence de l'expression de la formulation au regard des unités de ses variables : il s'agit, à partir des attributs

"unité" des variables d'appliquer les règles des opérations sur les unités afin de vérifier la cohérence globale de l'expression ;

° entrée-interdites : cet attribut désigne les variables (instances de la classe "Variable") qui sont jugées par l'Up comme ne pouvant pas être utilisées en entrées du modèle au risque d'avoir un problème "mal posé" ;

° type-d'implémentation : cet attribut monovalué et typé prend une valeur de l'ensemble {méthode, module, monolothique}. Il indique si la formulation du modèle est implémentée sous forme de méthode associée à la classe ou sous forme d'une subroutine dans un langage procédural (module) ou si elle est implémentée dans un langage procédural comme une partie d'une formulation globale (monolothique). Dans ce dernier cas, un démon attaché à l'attribut interdit la modification de la formulation du modèle indépendamment des autres modèles auxquels il est associé.

La figure III.9 représente un exemple d'instanciation de la classe "Modèle" et des classes "Formulation" et "Variable" associées pour la modélisation de la conduction thermique dans une paroi en régime permanent. Il est à noter que les conditions associées au modèle ne sont pas spécifier lors de la création de l'instance puisque l'attribut "type" des variables n'est pas encore spécifié. C'est lors de l'utilisation du modèle dans un assemblage que cet attribut est renseigné pour chacune des variables par affectation d'une des valeurs de l'ensemble {paramètre, entrée, sortie, dérivée}.

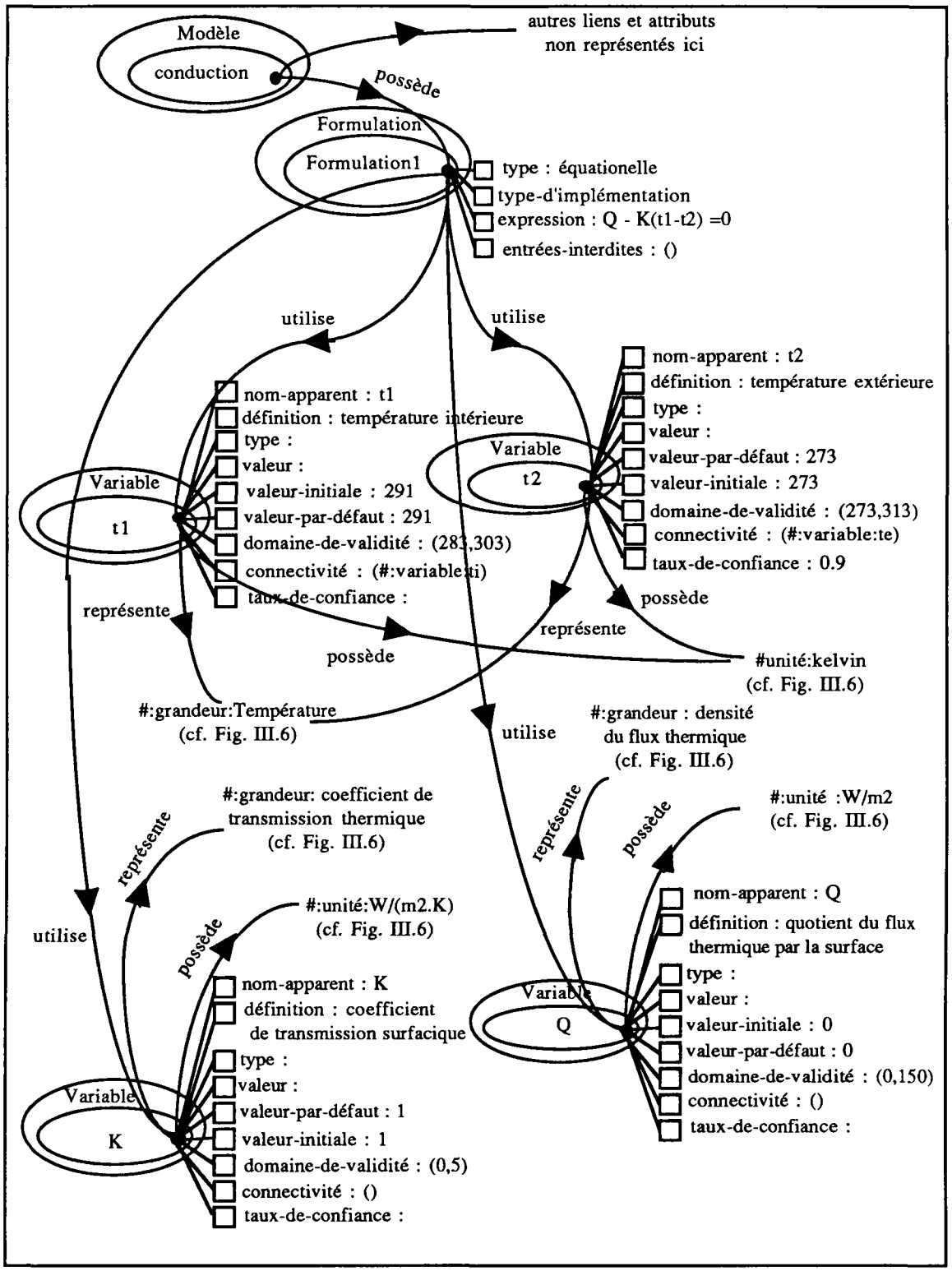


Figure III.9. Exemple d'instanciation des classes "Formulation" et "Variable".

3.2.14. Classe "Hypothèse"

Les hypothèses liées à un modèle sont des suppositions qualitatives et/ou quantitatives qui définissent le domaine de validité du modèle et les couplages qu'il peut avoir avec d'autres modèles. Les hypothèses peuvent être relatives :

° aux phénomènes traités par le modèle. Les hypothèses relatives aux phénomènes d'un modèle sont de deux types :

  - celles qui indiquent qu'un phénomène est pris en compte par le modèle (par exemple, le modèle tient compte des déperditions par le sol) ;

  - celles qui indiquent qu'un phénomène n'est pas pris en compte par le modèle (par exemple, le fluide est supposé incompressible).

° aux objets ou composants physiques représentés par le modèle (par exemple : la paroi est supposée homogène et isotrope) ;

° à des contraintes numériques sur certaines variables (internes ou non) du modèle (par exemple : la température intérieure est supposée constante).

Les hypothèses peuvent être reliées entre elles par des relations. Ces relations sont :

° est équivalente à : qui déclare que deux hypothèses sont équivalentes ce qui permet de donner des formulations différentes à la même hypothèse (par exemple, l'hypothèse "négliger la convection" est équivalente à "transferts convectifs nuls") ;

° implique : un modèle qui inclut une hypothèse H qui en implique des hypothèses Hi réagit comme s'il incluait les hypothèses Hi (par exemple, l'hypothèse "milieu homogène" implique, entre autre , l'hypothèse "transferts convectifs nuls") ;

° est contradictoire : certaines hypothèses peuvent être contradictoires (par exemple, les hypothèses "régime permanent" et "régime dynamique" sont contradictoires).

Au sein de l'ESI, la modélisation des hypothèses est réalisée afin de permettre de rattacher une hypothèse à un phénomène ou à un objet et de décrire les relations entre les hypothèses. Les hypothèses reproduisant des contraintes numériques sur des variables sont traitées, au niveau de la classe "Variable", à l'aide du domaine de validité associé à toutes les variables du modèle (cf.3.2.6). Le domaine de validité peut être réduit à une valeur pour décrire que la variable est supposée constante.

Une instance de la classe "Modèle" peut inclure plusieurs instances de la classe "Hypothèse" (à condition que ces hypothèses ne soient pas contradictoires). Les attributs de la classe sont :

° est-associée-phénomène : une instance de la classe "Phénomène" qui définit le phénomène physique associé à l'hypothèse ;

° est-associée-objet : une instance de la classe "Composant-technologique" qui définit l'objet physique associé à l'hypothèse ;

° est-équivalente : liste d'instances de la classe "Hypothèse" qui comporte les hypothèses jugées par l'auteur du modèle comme équivalentes à l'hypothèse en cours de description ;

° est-déduite : liste d'instances de la classe "Hypothèse" qui comporte les hypothèses jugées par l'auteur du modèle comme déduites de l'hypothèse en cours de description ;

° est-contradictoire : liste d'instances de la classe "Hypothèse" qui comporte les hypothèses jugées par l'auteur du modèle comme contradictoires à l'hypothèse en cours de description.

3.2.15. Classes "Lien-orienté", "Lien-non-orienté" et "Connexion"

Dans un assemblage, les couplages entre les modèles sont représentés par des liens qui peuvent être orientés ou non. Les classes "Lien-orienté" et "Lien-non-orienté" sont décrites comme des sous-classe de "Présentation" afin qu'elles héritent des caractéristisques graphiques pour leur affichage à l'écran.

Les attributs de la classe "Lien-orienté" sont les suivants :

° modèle-amont : attribut monovalué et typé qui prend comme valeur une instance de la classe "Modèle". Il représente le modèle d'où part la liaison orientée ;

° modèle-aval : attribut monovalué et typé qui prend comme valeur une instance de la classe "Modèle". Il représente le modèle où arrive la liaison orientée.

L'attribut de la classe "Lien-non-orienté" est :

° modèles-associés : attribut multivalué et typé qui prend comme valeur une liste d'instances de la classe "Modèle".

Dans un assemblage, outre les liens entre les modèles, il peut exister des connexions entre les variables des modèles. Ces connexions reproduisent des flux d'information, de matière ou d'énergie entre les modèles. Elles sont nécessairement orientées. La classe "Connexion" est décrite comme une sous-classe de "Lien-orienté" afin qu'elle hérite des attributs "modèle-amont" et "modèle-aval". Elle est aussi une sous classe de "Présentation" afin qu'elle hérite des caractéristisques graphiques pour son affichage à l'écran. Les attributs spécifiques à la classe "Connexion" :

variable-amont : attribut monovalué et typé qui prend comme valeur une instance de la classe "Variable" ;

variable-aval : attribut monovalué et typé qui prend comme valeur une instance de la classe "Variable".

Des démons rattachés aux attributs "variable-amont" et "variable-aval" vérifient que les instances de la classe "Variable" affectées aux attributs appartiennent respectivement au modèle-amont et au modèle-aval.

3.2.16. Classe "Méthode-de-résolution"

La résolution du modèle défini par un ensemble d'équations algébro-différentielle (sous une forme implicite) associé à des conditions initiales et/ou des conditions au limites nécessite généralement des moyens numériques (la résolution formelle n'étant pas possible dans la majorité des cas).

La résolution numérique peut apparaître comme une tâche annexe dans le processus de modélisation/simulation. Néanmoins, le choix de la méthode de résolution adaptée à un projet de simulation se fait en fonction des connaissances physiques associées aux modèles composant le projet.

Tandis que la majorité des codes de simulation existants ne prévoit qu'une méthode de résolution (généralement noyée dans les lignes de codes qui décrivent les modèles), d'autres donnent à l'utilisateur la possibilité de choisir dans une liste prédéfinie de méthodes de résolution, celle qui paraît la plus adaptée à un projet de simulation.

Par exemple : TRNSYS v13.1 donne à l'utilisateur la possibilité de choisir une des trois méthodes suivantes : Euler modifiée, Prédicteur-correcteur du $2^{ième}$ ordre, Prédicteur-correcteur du $4^{ième}$ ordre.

L'information sur la méthode de résolution associée à un projet de simulation est inclue au sein de l'ESI générique afin :

° d'être transmise vers les codes de simulation qui prévoient le choix de la méthode de résolution ;

° de permettre à l'auteur des modèles d'y associer, en fonction des connaissances physiques, une méthode de solution adaptée.

La méthode de résolution numérique est définie au sein de l'ESI générique comme une classe afin de permettre sa spécialisation au sein des ESI spécifiques. Elle est associée à la classe "Projet" et à la classe "Modèle".

L'attribut unique de la classe "Méthode de résolution" est :

° méthode : cet attribut est multivalué et typé. Il contient les noms des méthodes de résolution numériques. Il prend ses valeurs dans l'ensemble {différences finies, Crank-Nicholson, Newton-Raphson, facteurs de réponse, Euler, Exponentielle de matrice}. Cet ensemble n'est pas exhaustif est peut être enrichi par le concepteur d'application dans les ESI spécifiques.

3.2.17. Classe "Modèle-de-temps"

La gestion du temps de la simulation est pris en charge par les différents codes de simulation : ces codes maintiennent une "horloge" qui leur permet, après avoir résolu les systèmes d'équations relatifs à un assemblage de modèles dans un instant donné (par itérations jusqu'à la convergence du système), d'incrémenter la variable qui représente le temps dans les équations avant d'itérer à nouveau.

Afin que des codes de simulation différents puissent communiquer entre eux à travers l'ESI, il est indispensable de maintenir une "horloge" commune à ces codes : il s'agit d'une information unique et cohérente (par rapport aux horloge de chacun des codes de simulation) relative à l'évolution du temps dans le durée de simulation.

Cette horloge est représentée par la classe "Modèle-de-temps" : lors du déroulement d'une simulation mettant en jeu des codes différents, un de ces codes, après avoir incrémenté la variable temps dans ses équations, envoi la nouvelle valeur de la variable temps à l'ESI qui se charge de la rendre accessible aux autres codes.

Dans le cas où les codes de simulation n'utilisent pas le même pas de temps (par exemple, au cas où un des codes utilise un pas de temps constant et un autre utilise un pas de temps variable), c'est le plus petit pas de temps qui est retenu.

La classe "Modèle-de-temps" possède les attributs :

° heure : prend une valeur décrivant l'heure considérée au cours de la simulation Cette valeur est mise à jour à chaque pas de temps. Elle est représentée dans le format des heures décrit dans l'Annexe 1.

° date : prend une valeur décrivant la date considérée au cours de la simulation. Cette valeur est mise à jour à chaque pas de temps. Elle est représentée dans le format des dates décrit dans l'Annexe 1.

A ces deux attributs est rattaché un démon qui est déclenché suite à tout accès en écriture. Le démon se charge de répercuter les modifications des valeurs des deux attributs vers les différents codes de simulation de l'ESI multi-évaluateurs.

3.2.18. Classe "Utilisateur"

La modèle de l'utilisateur est une représentation de ce qu'on suppose être l'état des connaissances de l'utilisateur. L'utilisation d'un modèle de l'utilisateur au sein de l'ESI permet de réaliser un dialogue Homme/Machine qui s'adapte aux connaissances et aux références de ses utilisateurs.

Ces connaissances sont de deux types :

° des connaissances relatives au domaine traité ;

° des connaissances relatives à l'outil informatique.

La modélisation des connaissances relatives au domaine est un travail complexe qui nécessite la prise en compte :

° du niveau d'acquisition des concepts du domaine (connaissances factuelles) ainsi que des erreurs d'interprétation éventuelles de certains concepts ;

° des stratégies de raisonnement suivies (connaissances opératoires).

A l'heure actuelle, il existe quelques modèles opérationnels des connaissances de l'utilisateur dans des domaines scolaires (algèbre, arithmétique...). Dans des domaines plus complexes, des travaux de recherche sont en cours mais il n'existe pas de modèles opérationnels : un des principaux points de blocage est la compréhension des mécanismes d'évolution de l'expertise dans un domaine où les connaissances sont peu formalisées.

Le domaine de la modélisation/simulation est un domaine complexe où sont imbriqués du savoir et du savoir-faire : les connaissances sont généralement incomplètes et/ou incertaines et les stratégies de raisonnement sont essentiellement basées sur l'expérience. Par conséquent, une modélisation des connaissances de l'utilisateur dans

ce domaine est difficile à formaliser. Nous proposons donc d'intégrer, au sein de l'ESI, une définition explicite du niveau des connaissances du domaine par le moyen des "catégories d'utilisateurs" (cf. § 2) : il s'agit de permettre à des utilisateurs considérés comme des experts du domaine (Up) de donner un jugement sur les connaissances d'autres utilisateurs (Ui et Uf) et de décider de la "promotion" de ces utilisateurs vers d'autres catégories (rappelons que chaque catégorie a accès à un ensemble de fonctionnalités et que seule la catégorie des Up a accès à toutes les fonctionnalités de l'ESI).

La modélisation des connaissances de l'utilisateur relatives à l'outil informatique a pour but de réaliser des interfaces H/M qui s'adaptent à chacun des utilisateurs et qui améliorent, par conséquent, la flexibilité et la coopérativité de l'outil. Cette modélisation doit comporter des informations relatives à l'utilisateur concernant d'une part sa maîtrise de l'outil informatique et, d'autre part, ses préférences et ses méthodes de travail.

Le modèle intégré au sein de l'ESI est un modèle comportemental : il s'agit d'un modèle qui évalue les modifications à apporter à l'interface H/M en fonction du comportement de l'utilisateur vis a vis de l'outil (durée d'utilisation, erreurs de manipulation...). Le modèle est un modèle empirique dont la formulation initiale est faite par le concepteur d'application et affinées au fur et à mesure.

Les attributs de la classe "Utilisateur" appartiennent à quatres classes différentes. On peut distinguer les attributs associés :

° à la gestion des "catégories d'utilisateurs" (cf. § a) ;

° aux préférences de l'utilisateur (cf. § b) ;

° au modèle de l'utilisateur relatif à la maîtrise de l'outil (cf. § c) ;

° à une session de travail (cf. §d).

a. Les attributs associés à la gestion des "catégories d'utilisateurs" sont :

° nom : cet attribut prend comme valeur une chaîne de caractères qui représente le nom de l'utilisateur ;

° mot de passe : cet attribut prend comme valeur une chaîne de caractères qui représente le mot de passe de l'utilisateur. Il est utilisé afin d'interdire des accès intempestifs à la zone de travail de l'utilisateur (par exemple, pour l'accès en écriture dans les bibliothèques de modèles et de projets) ;

° répertoire : la valeur de cet attribut définit le chemin d'accès au répertoire de travail de l'utilisateur. Il est utilisé pour la gestion des sauvegardes des fichiers ;

° appartient-catégorie : cet attribut monovalué et typé prend comme valeur une instance de la classe "Catégorie de l'utilisateur". Un démon rattaché à l'attribut ne permet l'accès en écriture à cet attribut (pour la modification de sa valeur) qu'aux Up.

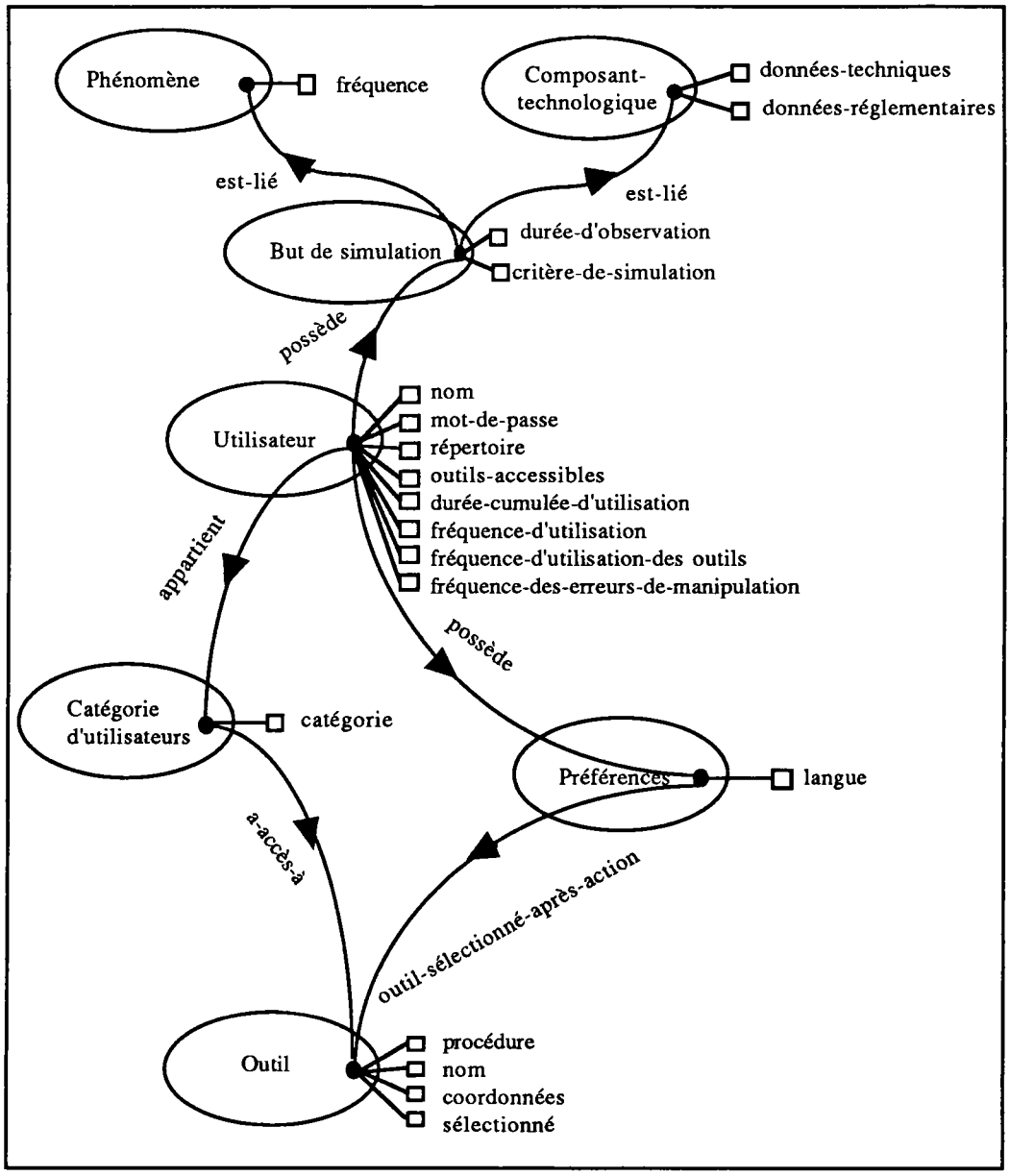


Figure III.10. Structure des classes "Utilisateur", "Préférences", "Catégorie d'utilisateur", "Outil", "But de simulation", "Composant technologique" et "Phénomène".

b. Les préférences de l'utilisateur sont représentées par une classe afin de permettre sa spécification dans les ESI spécialisés. Entre les classe "Utilisateur" et "Préférences" existe un lien représenté par l''attribut :

° possède-préférences : cet attribut prend comme valeur une instance de la classe "Préférences" (cf. §3.2.19).

c. Les attributs associés au modèle de l'utilisateur (relatif à la maîtrise de l'outil) sont les entrées du modèle. Les valeurs de ces entrées sont calculés par des démons associés aux attributs. Le modèle de l'utilisateur calcule alors, en fonction de ces entrées et des lois décrites par le concepteur d'application, les sorties. Ces sorties sont des attributs existants dont les valeurs seront modifiées par le modèle de l'utilisateur. Les entrées du modèle sont :

° durée-cumulée-d'utilisation : cet attribut prend comme valeur une chaîne de caractères dans le format des heures (cf. Annexe 1) qui représente, pour un utilisateur donné, le cumul des durées des sessions de travail. On émet l'hypothèse que la maîtrise de l'outil croit asymptotiquement en fonction de la durée d'utilisation. La valeur de l'asymptote correspond à la maîtrise parfaite de l'outil ;

° fréquence-d'utilisation : cet attribut prend une valeur numérique qui représente le quotient entre la durée d'utilisation et la durée de non-utilisation de l'ESI. Ce quotient est égal à X/P où X est la durée cumulée d'utilisation et P est la période totale.
Par exemple, pour une durée cumulée d'utilisation de 3h sur une période de 4 jours, la fréquence d'utilisation est de 3/(4*24) = 3%.

Cet attribut est utilisé pour faire la différence entre des utilisateurs "occasionnels" et des utilisateurs "réguliers" de l'ESI : on émet l'hypothèse que, pour une durée d'utilisation égale, le niveau de maîtrise de l'outil pour un utilisateur "occasionnel" est inférieure à celle d'un utilisateur "régulier" . Ceci est dû au fait que les connaissances sur l'utilisation d'un outil informatique (connaissances syntaxiques) sont stockées en mémoire à court terme où la dégradation avec le temps est rapide (cf. § 1.2.3 du Chapitre I).

° fréquence-d'utilisation-des-outils : cet attribut prend comme valeur une liste de doublets dont le premier élément est une instance de la classe "Outil" et le second est une durée d'utilisation. Cet attribut permet d'associer à chaque outil de l'ESI un "poids" d'utilisation (la métrique pour mesurer ce poids est la durée relative d'utilisation de l'outil par rapport à la durée d'utilisation globale). L'hypothèse sous-jacente à l'utilité de cet attribut est la suivante : l'utilisation de procédures complexes et optimisées reflète une maîtrise de l'outil informatique (cf. §2.1 du Chapitre 1). Par conséquent, en associant une fréquence d'utilisation aux outils permettant d'exécuter les procédures, on obtient une indication sur la maîtrise de l'ISE par l'utilisateur ;

° fréquence-des-erreurs-de-manipulation : à cet attribut est affectée une valeur numérique qui représente le quotient du nombre d'erreurs de manipulation sur la durée d'utilisation. Les erreurs de manipulation sont des actions qui sont annulées par l'utilisateur soit par l'infirmation de l'action à la suite d'un message de confirmation soit par annulation directe en utilisant la fonction "retour" (undo). On émet l'hypothèse que la fréquence des erreurs de manipulation est inversement proportionnelle à la maîtrise de l'ISE.

Les sorties du modèle de l'utilisateur sont :

° les attributs "image" des classes modèles, macro-modèles, projets et bibliothèques. L'utilisation de cet attribut par le modèle de l'utilisateur permet d'adapter cette présentation au niveau de connaissances de l'utilisateur : les "novices" ont une présentation à l'écran des entités plus détaillée et, au contraire, les "experts" ont une présentation plus abstraite ;

° l'attribut outils-accessibles de la classe "Utilisateur" : cet attribut prend comme valeur par défaut la liste des outils accessibles à la catégorie à laquelle appartient l'utilisateur. Cette valeur peut ensuite être modifiée par le modèle de l'utilisateur.

Par exemple : la liste des outils accessibles à la catégorie des Uf est la suivante : {instancier, effacer, éclater un macro-modèle, spécifier les paramètres, exécuter de la simulation}. La liste des outils accessibles à un utilisateur donné de la catégorie des Uf peut être : {instancier, effacer, spécifier les paramètres, exécuter de la simulation}.

3.2.19. Classe "Préférences"

La classe "Préférences" permet à l'utilisateur de spécifier explicitement certains paramètres de l'interface H/M de l'ESI. Elle est utilisée par la classe "Utilisateur". Ses attributs sont les suivants :

° outil-séléctionné-après-action : cet attribut permet de définir quel outil est sélectionné après l'exécution d'une action. Il prend comme une instance de la classe "Outil".
Par exemple : après l'utilisation de l'outil gomme pour supprimer un modèle, l'outil sélectionné peut être soit la gomme à nouveau soit un outil par défaut (l'outil d'instanciation par exemple);

° représentation-des-éléments-des-bibliothèques : les éléments des bibliothèques de modèles et de projets sont classés en hiérarchies. Ces hiérarchies peuvent être représenter soit sous forme d'arborescences soit sous forme de dossiers "à la Macintosh". Le choix entre ces deux représentations des mêmes entités est laissé à l'utilisateur. L'attribut prend alors comme valeur "arbres" ou "dossier" ;

° langue : cet attribut prend comme valeur un élément de l'ensemble {français, anglais, allemand...} qui détermine la langue dans laquelle les messages du système sont exprimés.

3.2.20. Classe "Catégorie d'utilisateur"

Cette classe permet de distinguer les utilisateurs d'un ESI en fonction de niveaux de leurs connaissances. Elle est définie comme une classe afin de faciliter sa spécialisation dans les ESI spécifiques. Ses attributs sont :

° catégorie : cet attribut monovalué et typé prend comme valeur un élément de l'ensemble {Up, Ui, Us} ;

° a-accès-à : cet attribut multivalué et typé prend comme valeur une liste d'instances de la classe "Outil". Cette liste répertorie les outils auxquels une catégorie d'utilisateur à accès.

3.2.21. Classe "Outil"

Cette classe décrit les outils par l'intermédiaire desquels les procédures/utilisateur de l'ESI peuvent être exécutées (cette classe ne concerne pas les procédures/système exécutées automatiquement par l'ESI). Elle est définie comme une sous-classe de "Présentation" puisqu'elle représente des entités visibles à l'écran.

Elle hérite par conséquent des attributs "nom", "coordonnées", "image" et "sélectionné" de la classe "Présentation". Elle possède un attribut spécifique :

° procédure : attribut monovalué et typé dont la valeur est un symbole qui pointe sur une procédure de l'ESI (par exemple, simuler, quitter...).

3.2.22. Classe "Composant-technologique"

Dans une optique d'une large diffusion de l'ESI hors du monde de la recherche, l'intégration des données relatives aux composants technologiques modélisés est fondamentale. Les composants technologiques peuvent être d'une part des types d'ouvrages et, d'autre part, des produits finis. Les données rattachées aux composants technologique sont le résultat de recherches dans un corpus technico-réglementaire de données avec utilisation des techniques hypermédia. Pour le bâtiment, ce corpus de données correspond, par exemple, au CD-REEF. Les données recherchées sont de deux types :

° les données réglementaires ; elles sont issues des travaux de normalisation et définissent les contraintes réglementaires sur l'utilisation d'un ouvrage dans un contexte donné (par exemple, étanchéité du gros œuvre en maçonnerie des toitures). Dans le cas du bâtiment ces données font l'objet, par exemple, de normes et de DTU ;

° les données techniques ; elles peuvent provenir d'une part des catalogues constructeurs et, d'autre part, des travaux de certification. Elles décrivent des données caractérisant le produit et son aptitude à l'emploi. Dans le cas du bâtiment ces données font l'objet, par exemple, des Avis Techniques.

La classe "Composant-technologique" est celle qui, au sein de l'ESI, gère le lien avec des bases documentaires : elle permet, suite à une demande d'information de la part de l'utilisateur concernant un composant donné, d'afficher les données réglementaires et et techniques disponibles dans une ou plusieurs bases documentaires concernant ce composant. Les choix techniques pour la réalisation des passerelles entre l'ESI et les bases techniques sont à effectuer pour chaque base.

Les attributs de la classe "Composant-technologique" sont les suivants :

° données-réglementaires. Cet attribut prend comme valeur une liste de références sur des textes réglementaires relatifs à un composant technologique ;

° données-techniques. Cet attribut prend comme valeur une liste de références sur des données techniques relatives à un composant technologique.

3.2.23. Classe "But-de-simulation"

Cette classe définit lors d'une session d'utilisation le but de simulation affiché par l'utilisateur. Ce but et défini par :

° le(s) composant(s) technologique(s) à étudier ;
° le(s) phénomène(s) à observer ;
° des critères de simulation ;
° une durée d'observation.

Les attributs de la classe "But de simulation" sont les suivants :

° est-lié-à-composant. Attribut multivalué et typé qui prend comme valeur une liste d'instances de la classe "Composant-technologique" ;

° est-lié-à-phénomène. Attribut multivalué et typé qui prend comme valeur une liste d'instances de la classe "Phénomène" ;

° durée-d'observation. Attribut monovalué et typé qui prend comme valeur un réel qui représente la durée d'observation des phénomènes dans le système de composants étudié ;

° crièтre-de-simulation. Attribut multivalué et typé qui prend comme valeur un ou plusieurs éléments d'une liste prédéfinie de critères de simulation. Cette liste est dressée par le concepteur d'application en fonction du domaine traité. Par exemple, pour le domaine du bâtiment, elle peut contenir {calcul de la consommation annuelle d'énergie, étude du confort, étude de la qualité d'air, optimisation de la consommation d'énergie au regard des stratégies de régulation...}.

3.2.24. Classe "Phénomène"

Les phénomènes représentent les événements que l'utilisateur désire observer au cours de la simulation.

La représentation des phénomènes par une classe permet de la spécialiser dans les ESI spécifiques.

Un modèle peut être spécifié comme adapté à l'étude de plusieurs phénomènes d'une part et que, d'autre part, l'utilisateur peut avoir un but de simulation qui combine plusieurs phénomènes à observer.

Les attributs de la classe "Phénomène" sont :

nom : cet attribut prend comme valeur un élément d'une liste de phénomènes. Afin de rendre sont exploitation plus évidente et afin de distinguer les différents niveaux des phénomènes, cette liste est présentée sous forme arborescente.

Par exemple :

- changement d'état
  - sublimation
  - évaporation
  - fusion
  - condensation

° fréquence : cet attribut définit le plus petit pas de temps dans lequel le phénomène observé change de caractéristiques. Cet attribut est utilisé par le modèle afin de définir la méthode de résolution numérique adaptée. Il peut être multivalué pour traiter le cas où la fréquence du phénomène n'est pas connue a priori de façon exacte. Il suffit alors de spécifier son intervalle d'encadrement.

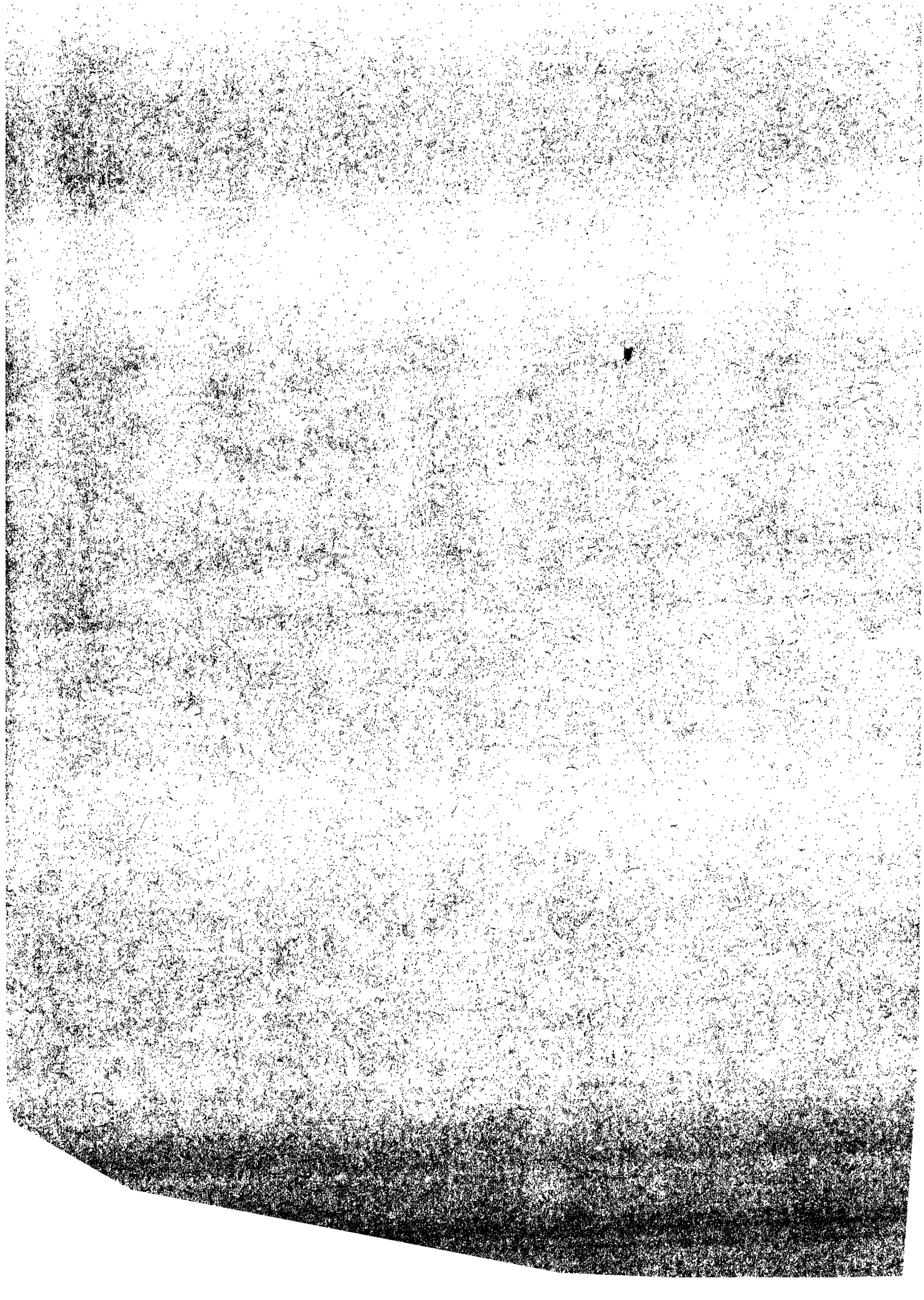
BILAN ET CONCLUSION DU CHAPITRE 3

Le but de ce chapitre est de proposer une modélisation d'un Environnement de Simulation Intelligent. En préalable, les concepts de base de la Représentation Orientée Objet, la méthode de spécification formelle suivie et la plate-forme logicielle utilisée pour l'implantation et la validation du modèle de l'ESI sont présentées.

Par ailleurs, une analyse de l'activité menée au sein d'une équipe travaillant dans le domaine de la modélisation / simulation a permis de distinguer quatre catégories d'utilisateurs :

- le concepteur d'applications qui décrit, pour un secteur d'activité donné, les entités manipulées et les modalités du dialogue H/M ;
- l'utilisateur principal qui constitue des bibliothèques de modèles représentant des systèmes et/ou des composants technologiques du secteur d'activité ;
- l'utilisateur intermédiaire qui se base sur les bibliothèques existantes afin de constituer des assemblages de modèles associés à des données de simulation ;
- l'utilisateur final qui exploite les assemblages prédéfinis afin de réaliser des études paramétriques.

Le modèle de l'ESI générique est ensuite détaillé. Ce modèle constitue la partie symbolique d'un environnement de simulation qui sert de base, d'une part, à un ensemble d'opérateurs sémantiques (vérification de la cohérence, modélisation du raisonnement, assistance à l'utilisateur...), et, d'autre part, à des traducteurs vers des codes de simulation qui prennent en charge la partie relative à la résolution numérique.



CHAPITRE IV : VALIDATION - PERSPECTIVES

INTRODUCTION

Dans ce chapitre, le modèle proposé de l'ESI générique (cf. § chapitre III) est utilisé pour la réalisation d'un prototype d'application opérationnelle. Ceci a pour but de tester l'opérationnalité du modèle proposé et de montrer l'enchaînement des différentes étapes nécessaires pour le développement d'un telle application.

Une application opérationnelle telle qu'elle a été définie (cf. § III.3) est un ESI adapté à un secteur d'activité, enrichi par des modèles mathématiques et intégrant un ou plusieurs codes de simulation.

Le secteur d'activité choisi pour le prototype décrit ci-après est le secteur du bâtiment. Les modèles mathématiques saisis sont ceux qui traitent du comportement thermique d'une zone. Le code de simulation utilisé est TRNSYS [TRN90].

1. REALISATION D'UN ESI SPECIALISE : ESI-BATIMENT

La réalisation d'un ESI spécialisé dans le secteur du bâtiment nécessite d'intégrer, au sein de l'ESI générique, une description (sous forme de classes et de relations entre les classes) du système technologique "bâtiment". Le modèle utilisé pour cette description est le modèle développé au CSTB et issu de la tâche IDM (Integrated Data Model) du projet européen COMBINE (programme JOULE de la DGXII des communautés européennes - [DEL92]). Ce modèle est aussi implémenté au sein de la plate-forme MIPS [PBD92].

La liaison entre l'ESI et l'IDM se fait à travers la classe "Composant technologique" de l'ESI : il s'agit de rattacher à cette classe la hiérarchie des classes de l'IDM. Les classes liées à la classe "Composant technologique" sont alors directement associées à la hiérarchie des classes de l'IDM. Par exemple, l'attribut "modélise-composant" de la classe "Modèle" qui implémente le lien entre cette classe et la classe "composant technologique" aura comme valeur un pointeur sur la hiérarchie de l'IDM. Ceci permet de rattacher chaque modèle de l'ESI au composant technologique qu'il modélise et, par conséquent, de faciliter le partage des données entre l'ESI et l'IDM (cf. Figure IV.1). Le partage des données est réalisé entre d'une part les attributs de la classe "Modèle" et, d'autre part, les attributs de la classe sélectionnée dans la hiérarchie des classes de l'IDM. Les attributs sont associés entre différentes classes en fonction de leurs noms.

Le partage de données est possible dans deux sens :

° les données sont saisies directement (ou à travers un module de CAO) au niveau de l'IDM et certaines d'entre elles sont affectées aux attributs des classes de l'ESI. Ce type de partage de données répond au besoin d'évaluation d'un projet de bâtiment existant ou en cours de conception suivant des critères techniques. Par exemple, il permet à un architecte ou un maître d'œuvre d'évaluer un Avant Projet Sommaire du point de vue des performances thermiques ;

° les données sont saisies directement au sein de l'ESI et certaines d'entre elles sont répercutées au niveau de l'IDM. Ce type de partage de données permet de mettre à la disposition des différents intervenants dans un projet de bâtiment certaines données saisies pour une évaluation technique spécifique. Par exemple, les données saisies pour l'évaluation thermique sont mis à la disposition d'un acousticien. Ceci permet d'une part d'alléger le travail de saisie des données pour des évaluations techniques ultérieures et, d'autre part, d'assurer la cohérence des données utilisées dans des évaluations techniques différentes.

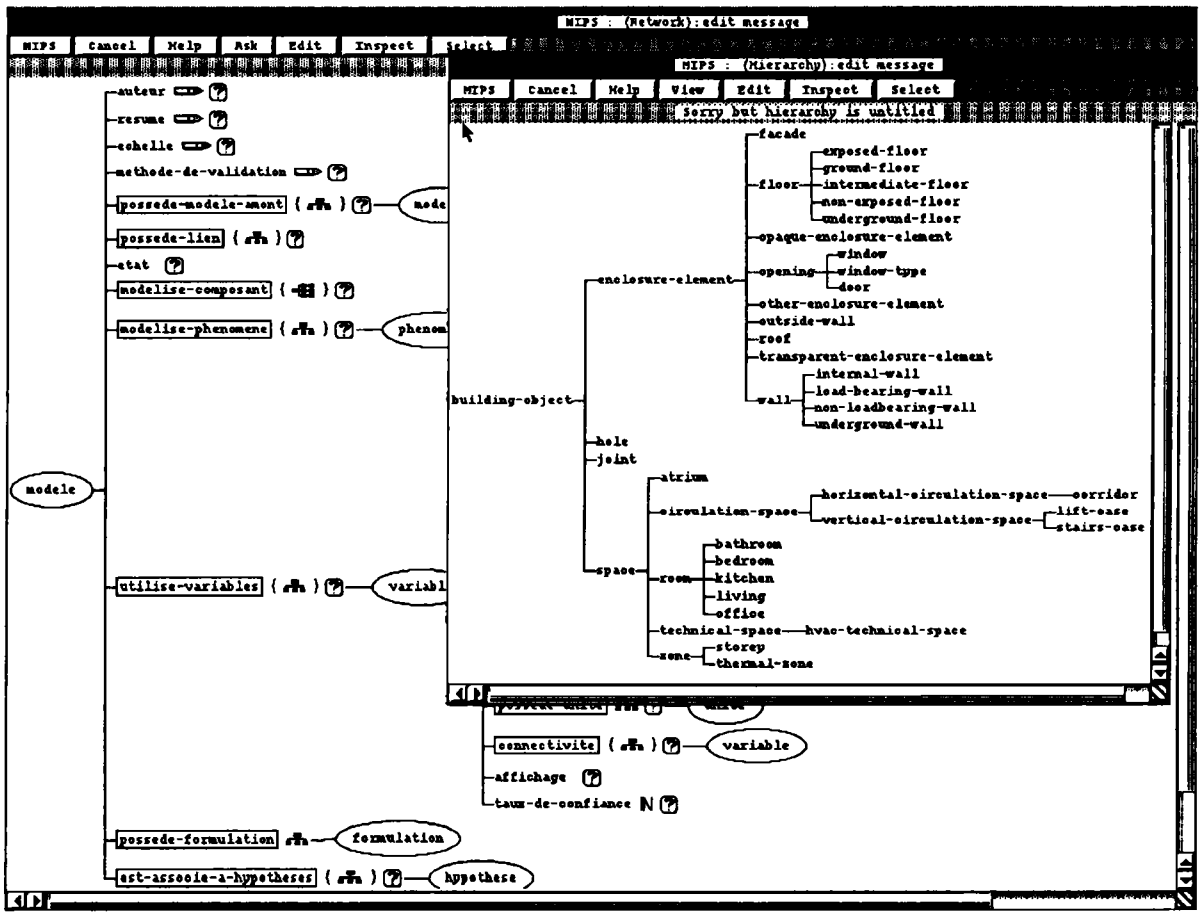


Figure IV.1. Structure des classes de l'ESI spécialisé dans le secteur du bâtiment : lien entre le modèle de l'ESI générique et le modèle de l'IDM.

2. EXEMPLE D'APPLICATION OPERERATIONELLE : SIMULATION D'UNE ZONE AVEC TRNSYS

L'insertion de bibliothèques de modèles associées à des codes de simulation dans la hiérarchie des classes de l'ESI spécialisé permet d'obtenir des applications opérationnelles. L'exemple décrit ci-après traite de la modélisation thermique d'une zone en utilisant TRNSYS comme code de simulation. TRNSYS est un code de simulation développé à l'Université de Wisconsin-Madison. Il est caractérisé par sa modularité qui lui permet d'intégrer facilement de nouveaux modèles : chaque modèle est représenté par une subroutine appelée module. Sa bibliothèque standard est orientée vers les calculs thermiques des bâtiments et de leurs équipements.

L'utilisation dans un ESI d'un code de simulation particulier nécessite le développement d'un traducteur entre l'ESI et ce code de simulation. Il s'agit d'une fonction qui récupère les valeurs des attributs des modèles utilisés dans un projet de simulation et génère un fichier d'entrée pour le code de simulation suivant la syntaxe adaptée. Un tel traducteur à été développé afin de permettre l'utilisation de TRNSYS dans cet exemple. D'autres traducteurs vers d'autres codes de simulation peuvent être développé en se basant sur le même modèle central de l'ESI générique. Par la suite, les instances des classes décrites au sein de l'ESI et utilisées afin de générer le fichier d'entrée pour TRNSYS sont explicitées et l'échange de données avec des instances de classes de l'IDM est détaillé.

2.1. DESCRIPTIF DU MACRO-MODELE "ZONE-DETAILLEE"

Le modèle utilisé correspond au TYPE 19 de la bibliothèque standard de TRNSYS. Il permet de calculer les charges de chauffage et de climatisation dans une zone. Les parois, plancher, toit, fenêtres et portes sont inclus dans le modèle ; les parois sont décrites par des fonctions de transfert.
Puisque ce modèle de zone regroupe la description de composants technologiques différents (portes, fenêtres, parois...), il est considéré dans l'ESI comme un assemblage de modèles élémentaires et, par conséquent, décrit comme une instance de la classe "Macro". Chacun des modèles élémentaires composant ce macro-modèle est décrit comme une instance de la classe "Modèle". Puisque la classe "Macro" est liée à la classe "Modèle" par une relation "est-composé-de" (cf.§3.2.5), l'instance "Zone-détaillée" de la classe "Macro" possède des liens de type "est-composé-de" avec les instances de la classe "Modèle" (cf. Figure IV.2).

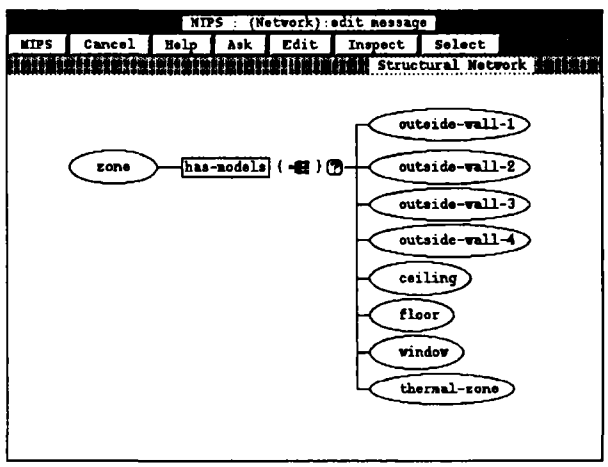


Figure IV.2. Structure de l'instance "Zone-détaillée" de la classe "Macro" de l'ESI.

2.1.2. Descriptif du modèle "Zone"

Le modèle de "Zone" utilisé par le macro-modèle "Zone-détaillée" permet de définir d'une part les données intrinsèques de la zone (conditions climatiques,volume, inertie...) et, d'autre part, des données relatives à l'usage de la zone (nombre d'occupants, taux de renouvellement d'air, température de consigne...- cf. Figure IV.3). La modélisation du transfert de chaleur entre les éléments de la zone utilise l'équation matricielle suivante :

$$[Z_{i,j}]\,[T_{s,i}] = [X_i] \qquad \text{(Eq. IV.1)}$$

où

Ts,i représente d'une part les températures de surfaces des éléments i et, d'autre part, la température d'air pour i=n+1 ;
Xi regroupe les facteurs dont la variation a des incidences sur les Ts,i ;
Zi,j sont les coefficients de transfert entre les éléments i et j.

La méthode de résolution numérique utilisée est la méthode d'inversion de matrice.

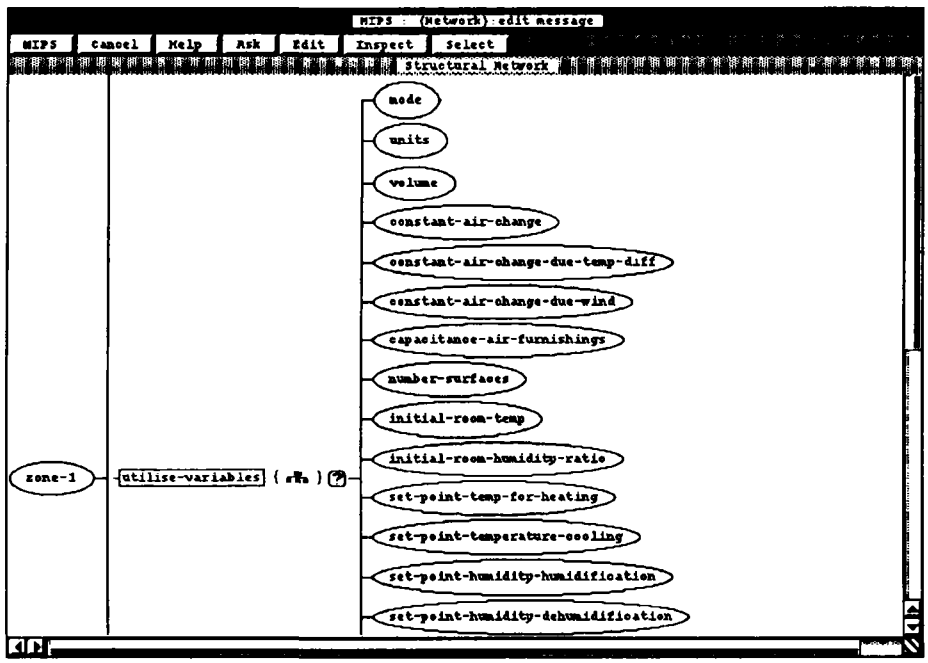


Figure IV.3. Paramètres de l'instance "Zone" de la classe "Modèle" de l'ESI.

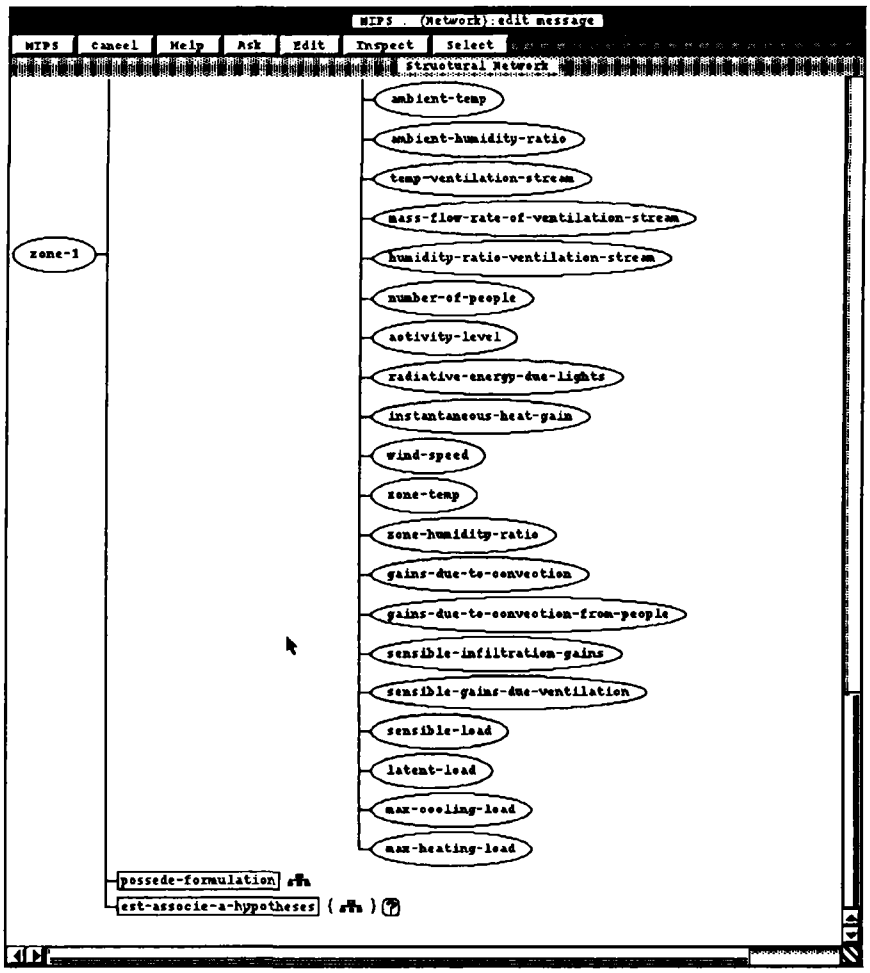


Figure IV.4. Partie de la structure de l'instance "Zone" de la classe "Modèle" de l'ESI : entrées et sorties du modèle.

2.1.3. Descriptif du modèle "Parois-opaques"

Le modèle des parois opaques sert à la description des murs extérieurs et des cloisons ainsi qu'à celle du plancher et du plafond. Il est basé sur une description par des fonctions de transfert dont les coefficients proviennent du "ASHRAE Handbook of Fundamentals".

L'echange de chaleur instantané entre la zone et les murs extérieurs est modélisé suivant la fonction de transfert suivante :

$$qi = S\ b_{h,i}\ T_{sa,i,h} - S\ c_{h,i}\ T_{eq,i,h} - S\ d_{h,i}\ q_{i,h} \qquad \text{(Eq. IV.2)}$$

Les coefficients b_h, c_h et d_h sont les coefficients de transfert pour la température ambiante (T_{sa}) , pour la température de la zone (T_{eq}) et pour les flux de chaleur (q). Le coefficient d'échange de rayonnement GLO est supposé linéaire et les surfaces supposées noires.

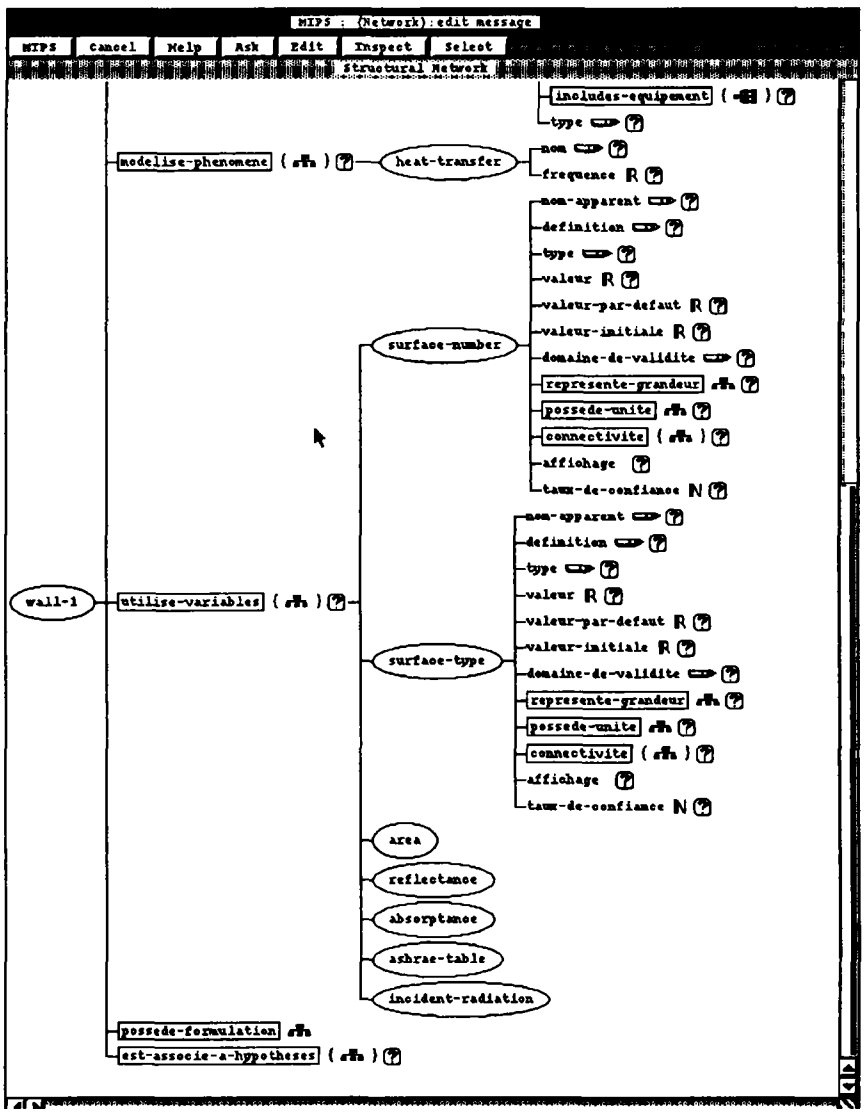


Figure IV.5. Structure d'une instance "Paroi-opaque" de la classe "Modèle" de l'ESI.

2.1.3. Descriptif du modèle "Paroi-vitrée"

Le modèle de paroi vitrée permet de calculer les apports solaires en fonction du rayonnement direct et de la transmittance de la paroi vitrée. L'échange de chaleur à travers la paroi vitrée est modélisé suivant l'équation :

$$Q_i = A_i U_{g,o,i} (T_a - T_{eq,i}) \qquad \text{(Eq. IV.3)}$$

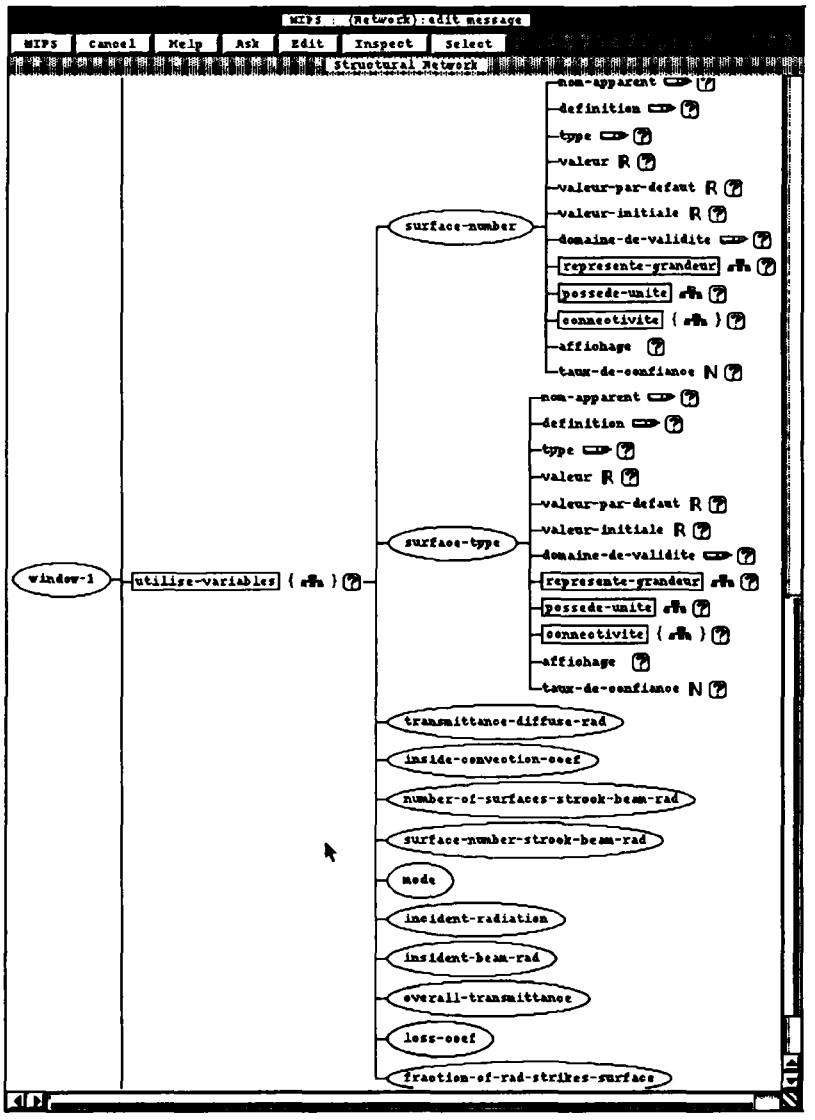


Figure IV.6. Structure de l'instance "Paroi-vitrée" de la classe "Modèle" de l'ESI.

2.2. DESCRIPTIF DES CLASSES UTILISEES DANS L'IDM

Au sein de l'IDM, les classes nécessaires à la réalisation de l'exemple présenté sont groupées dans les hiérarchies suivantes :

° "Space" (espace) : décrit un volume utilisable dans un bâtiment. Cette hiérarchie contient, entre autre, la classe "Zone" et sa sous-classe "Thermal-zone" ;

° "Enclosure-element" (élément d'enveloppe) : cette hiérarchie contient les classes permettant de décrire l'enveloppe du bâtiment. Elle contient notamment les classes qui décrivent les parois opaques et les parois vitrées de l'enveloppe.

2.2.1. Descriptif de la classe "Thermal zone"

La classe qui permet de représenter au mieux le modèle de zone utilisé dans TRNSYS est la classe "Thermal zone" de la hiérarchie "Space". Cette classe est une sous-classe de "Zone" et elle regroupe, en plus des attributs hérités des classes mères, les attributs nécessaires à une analyse thermique d'une zone. L'ensemble des attributs de la classe "Thermal zone" est représenté dans la figure IV.7.

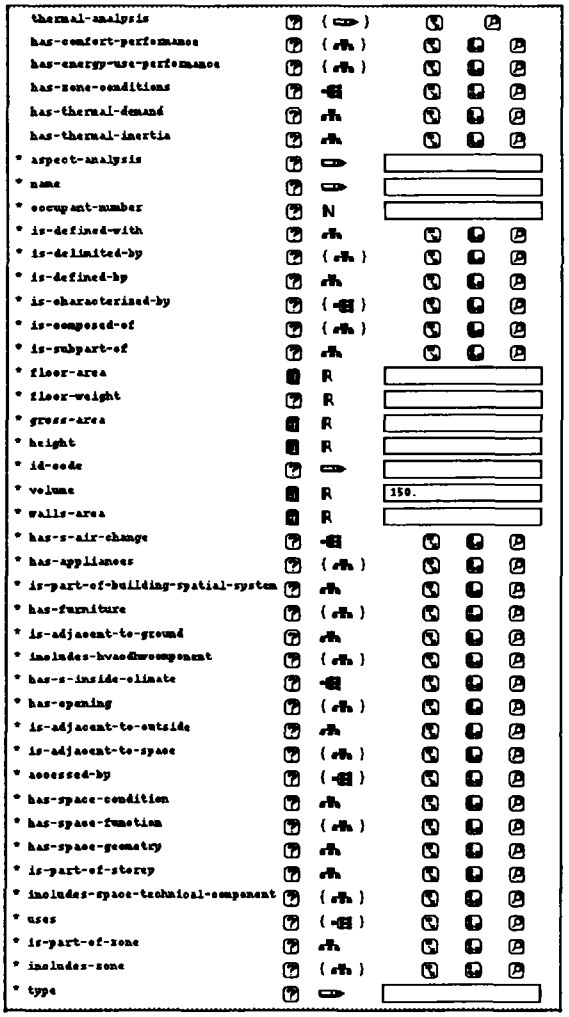


Figure IV.7. Structure de la classe "Thermal-zone" de l'IDM.

2.2.2. Descriptif de la classe "Wall"

Cette classe appartient à la hiérarchie "enclosure-element" et permet de décrire les parois internes et externes d'un bâtiment. Ces attributs sont représentés dans la figure IV.8.

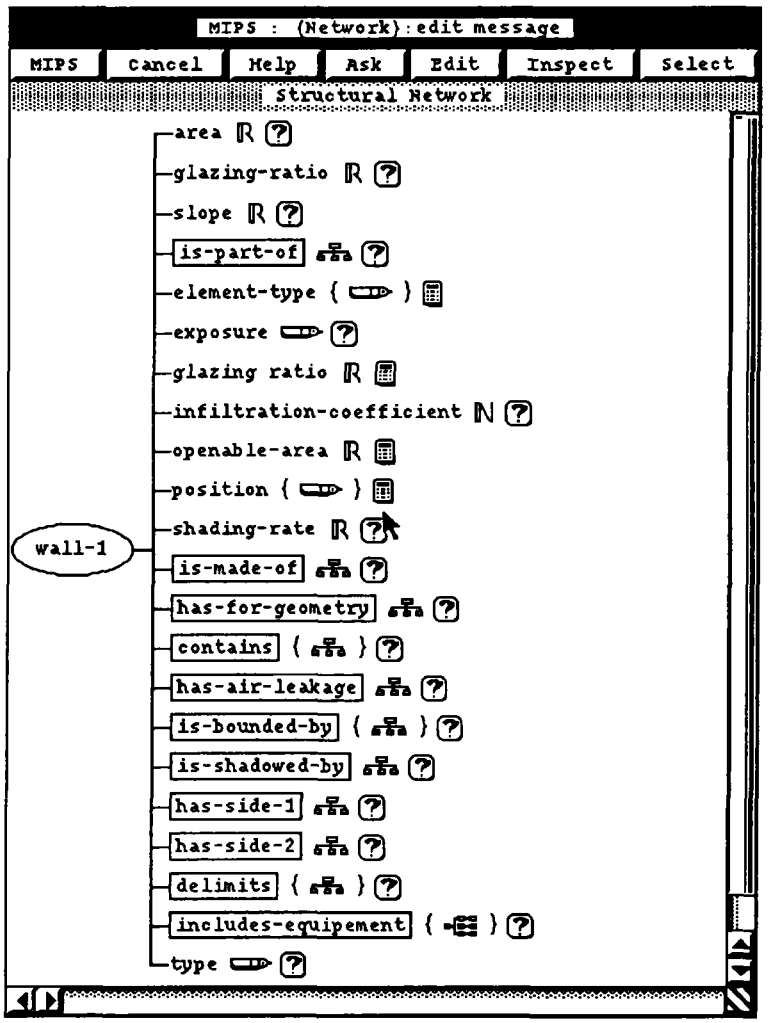


Figure IV.8. Structure de la classe "Wall" de l'IDM.

2.2.3. Descriptif de la classe "Window"

Cette classe appartient à la hiérarchie "enclosure-element". Elle permet de décrire les parois vitrée d'une enveloppe de bâtiment. Ces attributs sont représentés dans la figure IV.9.

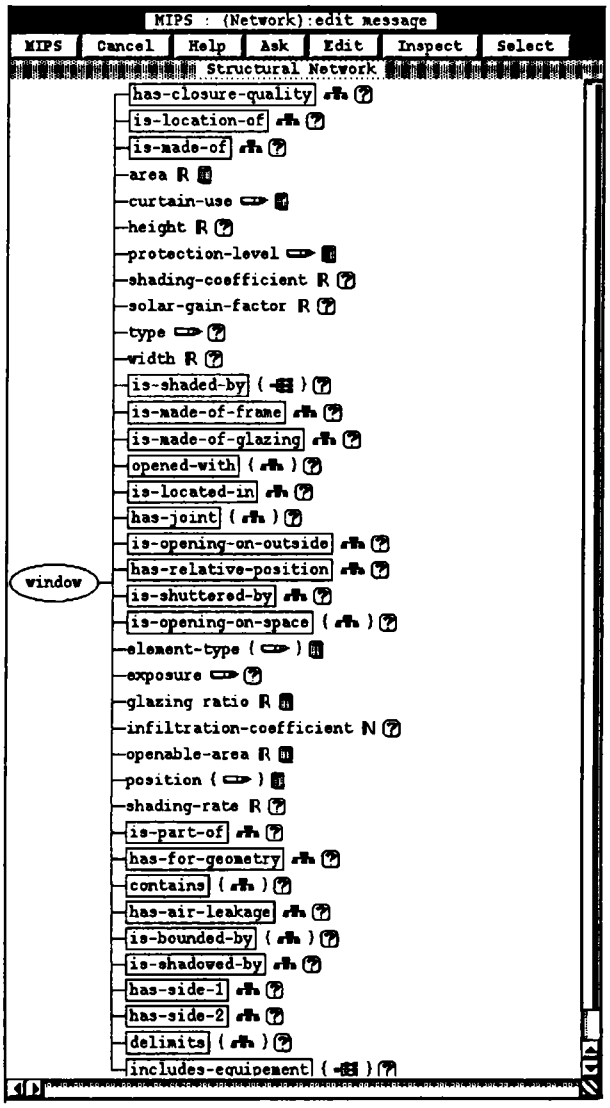


Figure IV.9. Structure de la classe "Window" de l'IDM.

2.5. ECHANGE DE DONNEES ENTRE ESI-IDM

Suite à la création d'instances des classes utilisées dans l'IDM et dans l'ESI, des liens entre ces instances peuvent être décrits et l'échange de données peut être réalisé.

Les liens sont basés sur l'attribut "modélise-composant" de la classe "Modèle" de l'ESI : dans le cas de l'ESI adapté au secteur du bâtiment, cet attribut va pointer sur la hiérarchie de l'IDM décrivant le bâtiment. Ceci permet à l'auteur d'un modèle dans l'ESI de désigner le composant technologique représenté par son modèle en séléctionnant la classe adaptée dans l'IDM.

Par exemple, à l'instance "zone-1" représentant le modèle de zone dans TRNSYS est associée, par l'intermédiaire de l'attribut "modélise-composant", l'instance "thermal-zone-1" de la classe "Thermal-zone" de l'IDM (cf. Figure IV.10). Un lien équivalent existe entre les instances "Paroi-opaque", "Paroi-vitrée" de l'ESI et, respectivement, les intances "Wall" et "Window" de l'IDM.

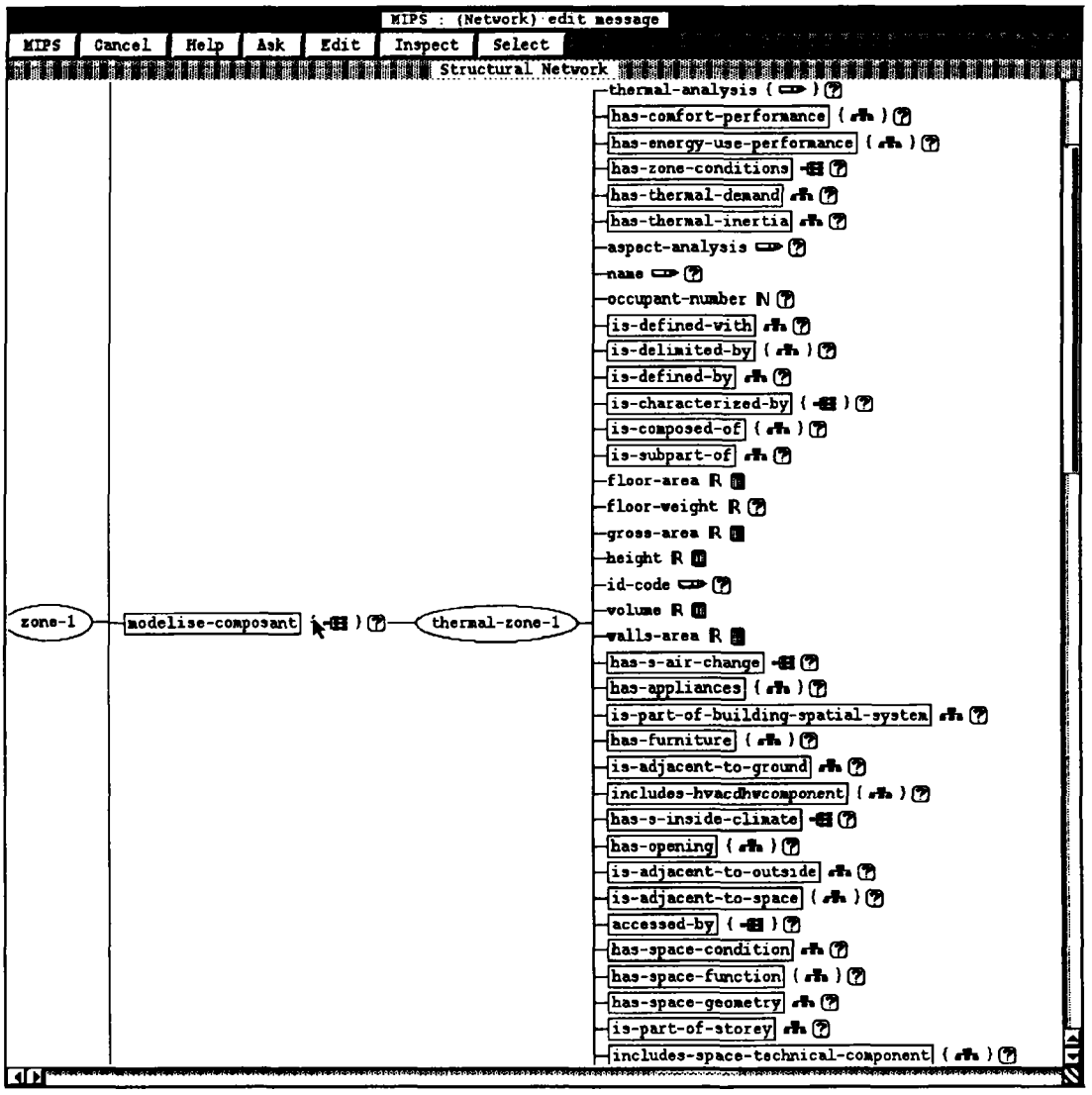


Figure IV.10. Lien entre les instances "zone-1" de l'ESI et "thermal-zone-1" de l'IDM.

L'échange de données entre l'ESI et l'IDM peut alors être réalisé par l'intermédiaire d'une fonction de requête. Il s'agit d'une fonction déclenchée par une action de l'utilisateur qui, à partir de l'instance de "Modèle" sélectionnée, parcourt la hiérarchie des classes de l'IDM afin de trouver, pour chacune des variables du modèle, un attribut portant le même nom que la variable. La valeur de l'attribut trouvé est alors affectée à l'attribut "valeur" de la variable du modèle. Au cas où cette variable est un paramètre, cette valeur est alors utilisée tout au long de la simulation. Au cas où il s'agit d'entrée ou de sortie, cette valeur est considérée comme une valeur initiale pour la simulation.

Par la suite, un exemple d'implantation d'une telle fonction de requête sera décrit. Il a été intégré au sein d'une application dédiée à l'étude du comportement thermique du bâtiment. Cette application (cf. Figure IV.11), détaillée dans un document de thèse qui sera soutenue au courant de l'année [EHa92], représente une première utilisation du modèle profond de l'ESI. D'autres applications sont en cours de développement.

Dans cette application, l'utilisateur réalise un assemblage de modèles en puisant dans les bibliothèques existantes. Cet assemblage contient les informations nécessaires pour exécuter la simulation : à chacun des modèles, représentés sous forme iconique, sont rattachés des paramètres. Les couplages entre les modèles sont représentés par des liens orientés.

La fonction de requête est rattachée à un outil spécifique de la boîte à outils. L'utilisateur sélectionne le modèle sur lequel portera la requête (dans la figure IV.11, ce modèle est celui d'une zone-thermique). La fonction de requête retrouve alors dans l'IDM toutes les instances qui peuvent correspondre au modèle sélectionné et les propose à l'utilisateur. Une fois que l'utilisateur sélectionne une des ces instances, la fonction de requête parcours la hiérarchie associée (cf. Figure IV.12) afin de retrouver des attributs portant les même noms que les paramètres du modèle. Les valeurs de ces attributs sont alors affectées aux paramètres associés.

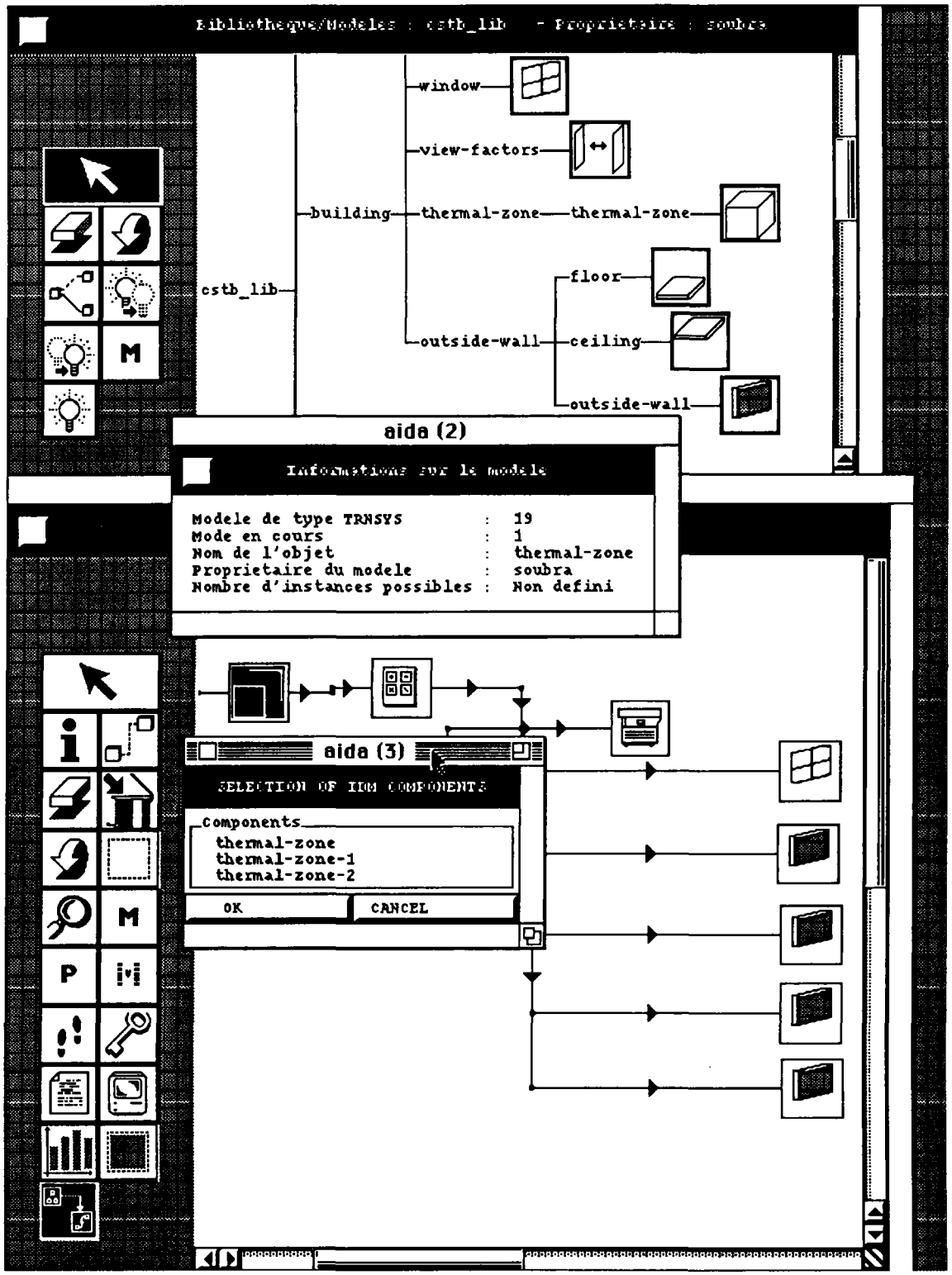


Figure IV.11. Requête sur les instances de "zone-thermique".

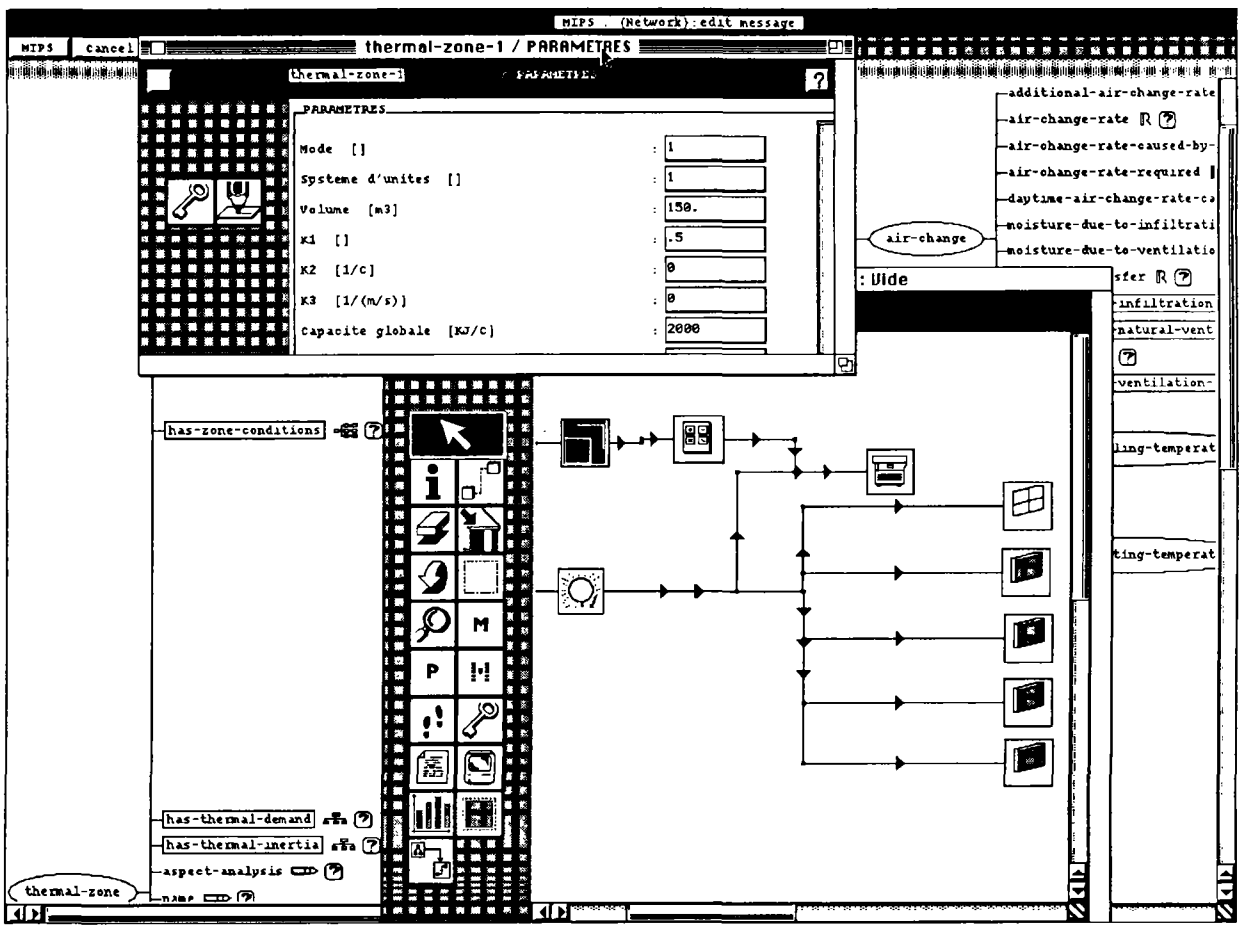


Figure IV.12. Partie de la hiérarchie parcourue lors de la requête basée sur "zone-thermique" et affectation des valeurs des paramètres.

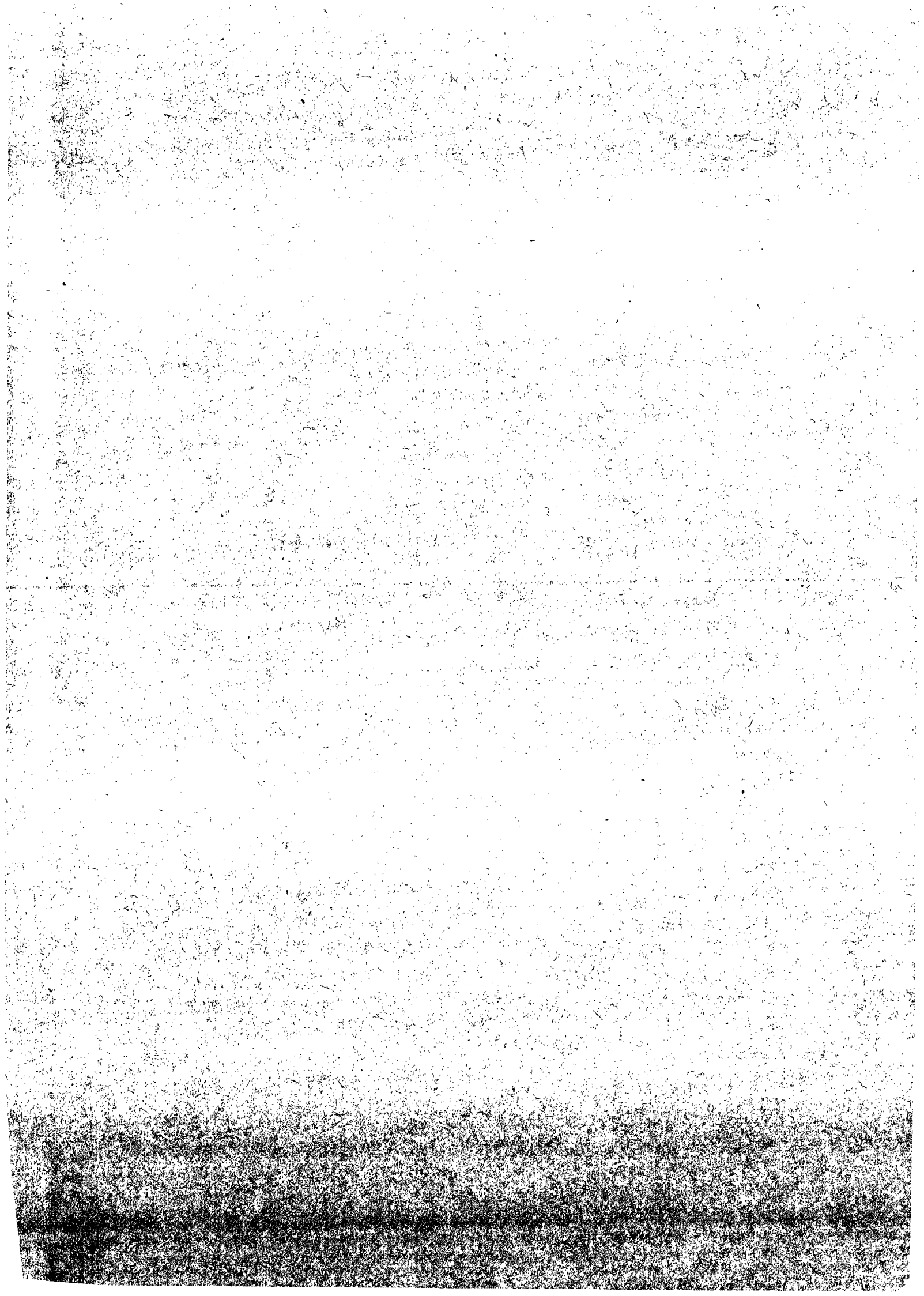
Pour l'exemple étudié, la requête a conduit à l'affectation de valeurs provenant de l'IDM aux variables suivantes :

Modèle	Variable renseignée
zone	Volume
	Initial room temperature
	Initial room humidity ratio
	Set point temperature for heating
	Set point temperature for cooling
	Set point humidity ratio for dehumidification
wall	area
window	area

Ces valeurs sont ensuite complétées au sein de l'ESI afin d'obtenir l'ensemble des valeurs nécessaires au lancement de la simulation avec TRNSYS. Il s'agit notamment des variables spécifiques à TRNSYS (mode, système d'unités...) et des variables dont les noms sont différents dans l'IDM et dans l'ESI (le taux de renouvellement d'air est nommé respectivement "constant air change per hour" et "air change rate" dans l'ESI et dans l'IDM).

BILAN ET CONCLUSION DU CHAPITRE 4

Le prototype d'application pour le secteur du bâtiment présenté a permis de montrer l'opérationnalité du modèle de l'ESI générique. Bien que notre objectif était purement fonctionnel, l'échange de données entre l'ESI et l'IDM n'est pas négligeable. Le développement d'un langage de requête dédié permettant de remplacer le critère de recherches basé sur le nom de l'attribut par un critère plus complexe prenant en compte la signification physique des variables améliorera substantiellement les performances du système. Ce langage de requête peut alors intégrer des vérifications sur les unités et des conversions automatiques des valeurs au regard du système d'unités utilisé ainsi qu'une paramétrisation par l'utilisateur de la profondeur de la requête.



CONLUSIONS GENERALES

Le travail de recherche présenté s'inscrit dans le contexte de la réflexion sur le développement d'environnements de simulation de nouvelle génération dans le secteur du bâtiment. Le point de départ est le constat suivant : contrairement à d'autres secteurs d'activité (électronique, espace...), les codes de simulation existants dans le secteur du bâtiment, principalement issus de la recherche, sont peu utilisés par les professionnels (bureaux d'étude, maître d'ouvrage, maître d'œuvre...). Ceci est principalement dû au fait que ces codes, malgré leur performances numériques et leur larges champs d'application, manquent d'un certain nombre de fonctionnalités symboliques afin que leur utilisation soit simple et productive.

Nous proposons donc une méthode pour le développement d'environnements hybrides : il s'agit d'environnements qui comportent, d'une part, une partie symbolique assurant un ensemble d'opérateurs sémantiques et, d'autre part, une partie numérique prenant en charge la résolution numérique proprement dite.

Cette méthode a l'avantage de permettre d'intégrer des codes de simulation existants et également d'implanter, de façon innovante, des modèles sous forme de méthodes rattachées aux classes afin de bénéficier des propriétés d'héritage de l'approche orientée objet.

Cet ensemble hybride désigné par Environnement de Simulation Intelligent (ESI) assure les fonctionnalités suivantes :

° l'intégration de codes de simulation différents traitant d'évaluations techniques dans des domaines variés (thermique, acoustique, qualité de l'air...) ;

° l'échange de données entre ces différents codes afin de faciliter le travail de saisie et d'assurer l'unicité des données d'un projet de simulation évalué par plusieurs codes ;

° l'archivage des modèles et leur partage entre différentes catégories d'utilisateurs.

Le modèle d'ESI proposé est générique : la description des classes qui le composent ne comporte pas de spécificité par rapport à un secteur particulier. Il peut être utilisé dans tout domaine où la simulation représente un moyen d'investigation du comportement de systèmes technologiques complexes soumis à des sollicitations. Il est néanmoins plus adapté à une approche par décomposition récurrente des systèmes.

Ce modèle d'ESI a ensuite été appliqué au secteur du bâtiment ce qui a permis de tester l'opérationnalité du modèle et de montrer l'enchaînement des étapes nécessaires pour le développement d'une application opérationnelle sur la base du modèle générique.

Le travail de recherche a permis de mettre en évidence deux axes d'évolution :

° le premier axe se propose d'étudier les possibilités de modélisation du raisonnement dans un environnement de simulation. Il devrait, à terme, conduire à un ensemble de règles de production et d'heuristiques qui formalise la connaissance sur les limites de l'utilisation des modèles. Cette connaissance peut être inhérente aux modèles ou issue du système technologique et/ou du domaine technologique étudié ;

° le second axe se propose d'investiguer plus en détail les possibilités de couplage entre des codes de simulation différents. Il s'agit d'utiliser le modèle de l'ESI générique comme superviseur d'une simulation faisant intervenir plusieurs codes de simulation. L'ESI pilote la simulation, s'assure de la convergence de chacun des codes et de la convergence globale. Dans le cas de non convergence l'ESI peut faire appel à des bases de règles afin définir les actions qu'il serait possible d'entreprendre.

LISTE DES FIGURES

REFERENCES BIBLIOGRAPHIQUES

[Bac76] BACHELARD. *Structure Dynamique des systèmes.* Coll. Recherches Interdisciplinaires. Paris. 1976.

[Bez86] J. BEZIVIN. *Smalltalk et prototypage.* Génie logiciel n°3. Janvier 1986. pp.16-22.

[BF81] A. BARR - E.A. FEIGENBAUM. *Le manuel de l'Intelligence Artificielle.* Eyrolles Ed. 1981.

[BHT85] C. BONNET - J.M. HOC - G. TIBERGHIEN. *Psychologie, intelligence artificielle et automatique.* Ed. Pierre Mardaga. Bruxelles. 1985.

[Bis86] A. BISSERET. *Towards computer - aided text production.* 3rd European Conference on Cognitive Ergonomics. Paris 15-19 September 1986.

[BKP76] P.A. BOBILLIER - B.C. KAHAN, A.E. PROBST. *Simulation with GPSS and GPSS V.* Prentice - Hall. Englewood Cliffs New Jersey. 1976.

[BO87] P. BELL - R. O'KEEFE. *Visual Interactive Simulation - History, recent developments and major issues.* Simulation. Septembre 1987. pp. 109-116.

[Boo88] G. BOOCH. *Ingéniérie du logiciel avec Ada, de la conception à la réalisation.* InterEditions. Paris. 1988.

[BDP91] E. BRISSON - P. DEBRAS. P. POYET. *A First Step towards an Intelligent Integrated Design System in the Building field.* CIB W78. The Computer Integrated Future. Eindhoven. 16-17 Septembre 1991.

[CMN83] S. CARD - T. MORAN - A. NEWELL. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates. 1983.

[Cou87a] J. COUTAZ. *The construction of user interfaces and the object paradigme.* Proceedings of the European Conference on Object Oriented Pradigme (ECOOP). 1987.

[Cou87b] J. COUTAZ. *Méthodes et outils pour l'implémentation des interfaces modernes.* Cours INRIA. 1987.

[CRR89] J.A. CLARKE - D.M. RANDAL - J. RUTHERFORD. *The Application of Intelligent Knowledge Based Systems in Building Design.* Repport for Grant GR/E/18018. 1989.

[Cla91] J. CLARKE. *The Energy Kernel System : The way ahead ?.* Building Environmental Performance 91. Canterbury. 10-11 Avril 1991.

[DPB92] P. DEBRAS - P. POYET - E. BRISSON. *Expert Systems and Documentary Databases.* Micro-computers in Civil Engineering, An International Journal - Elsevier Science Publishers. 1992. 10 pp.

[Ded86] C. DEDE. *A review and synthesis of recent research in intelligent computer - assisted instruction.* International Journal of Man - Machine Studies. n°24. 1986. pp. 329-353.

[DEL92] A.M. DUBOIS - J.C. ESCUDIE - Louis LARET. *COMBINE IDM / V3.3 / 920123.* CSTB. DPE/TTA/MGL/92-1264. 1992.

[Dub87] A.M. DUBOIS. *Le PROFORMA : mythes et réalités.* CSTB. TTA/MGL/787. 1987.

[Dub88] A.M. DUBOIS. *A draft of a component model library manager, the MODELOTHEQUE : analysis of an approach and first results.* 3 rd Int. Itern. Symp. Systems analysis and simulation. Berlin. Septembre 1988.

[EN81] D.W. EMBLEY - G. NAGY. *Behavioral aspects of text editors.* ACM Computing Surveys. 13, (1). March 1981. pp. 33-70.

[Fal87] P. FALZON. *Communication Homme-Homme et Interaction Homme-Machine.* Bulletin de Liaison de la Recherche en Informatique et en Automatique. 115/1987. pp. 13-21.

[FP90] M. FORNARINO - A.M. PINNA. *Un modèle objet logique et relationnel. Le langage OTHELO.* Thèse. Université de Nice. 1990.

[Flo83] C. FLOYD. *A systematic look at prototyping.* Proceedings of the working conference on prototyping. Namur. Octobre 1983.

[Für91] J.M. FÜRBRINGER. *Evaluation procedure using sensitivity analysis of model and measurments.* Document IEA. Annexe 23. Ottawa. 1991.

[FWC84] J.D. FOLEY - V.L. WALLACE - P. CHAN. *The human factors of computer graphics interaction techniques.* IEEE. Novembre 1984. pp. 13-47.

[FKK+89] J.D. FOLEY - W.C. KIM - S. KOVACEVIC. *Defining interfaces at a High Level of Abstraction.* IEEE. Janvier 1989. pp. 25-32.

[Git86] D.T. GITTINS. *Icon-based human-computer interaction.* International Journal of Man-Machine Studies. n°24. 1986. pp. 519-543.

[Gol84] A. GOLDBERG. *Smalltalk-80 : The interactive programming environment.* Addison Wesly Publications. 1984.

[Gou86] G. GOUARDERES. *Représentation et manipulation des connaissances dans le dialogue Homme-Machine en EAO.* Thèse. Université Paul Sabatier de TOULOUSE. 1986.

[EHa92] K. EL HASSAR. Conception d'un Environnement de Simulation Intelligent. Thèse. Ecole Nationale des Ponts et Chaussées. A paraître.

[HBR81] P. HAYES - E. BALL - R. REDDY. *Breaking the Man-Machine communucation barrier.* Computer magazine. 1981. pp. 19-30.

[Heb75] J.HEBENSTREIT. *Informatique et pédagogie.* Bulletin de Psychologie. XXVIII. 1975. pp.358 - 365.

[Heb88] J.HEBENSTREIT. *Simulation et pédagogie - Une rencontre du troisième type.* AFCET / INTERFACES. Mars 1988. pp.14 -18.

[Hol84] J.D. HOLLAN. *STEAMER : An Interative Inspectable Simulation Based Training System* . The AI Magazine. Summer 1984. pp 15-27.

[Hou88] G. HOUBART. *Des objets et des icônes pour l'interface utilisateur du système de conception graphique COGITO.* Actes MICAD. 1988.

[Hul86] J.M HULLOT. *Un générateur d'interfaces Homme-Machine.* Journées Langages Orientés Objet. 1986.

[HS89] W.D. HURLEY - J.L. SIBERT. *Modeling User Interface-Application Interactions.* IEEE. Janvier 1989. pp. 71-77.

[JL83] JOHNSON - LAIRD. *Mental Models.* Cambridge University Press. 1983.

[KeB83] W.A. KELLOG - T.J.BREEN. *Evaluating user and system models : applying scaling techniques to problems in Human-Computer Interaction.* Human Factors in Computing Systems-IV. J.M. Caroll and P.P. Tanner (Eds). ACM. North-Holland. Amsterdam. 1983.

[KB84] D.E. KIERAS - S. BOVAIR. *The role of a Mental Model in Learning to Operate a Device.* Cognitive Science. 8. 1984. pp. 255-273.

[Lah89] A. LAHELLEC. *Problématique du Raccordement de Modèles.* Séminaire spécialisé "Outils d'aide à la conception et la gestion". AFME - Sophia Antipolis. 1989.

[Lar88] L. LARET. *An attempt for adapted model generation methodology : example of application in building energy modelling.* 3 rd Int. Itern. Symp. Systems analysis and simulation. Berlin. Septembre 1988.

[LD88] L. LARET - A.M. DUBOIS. *Modélisation et simulation du fonctionnement thermique des bâtiments et des équipements associés.* CSTB. MGL/88-1031. 1988.

[Leg86] B. LEGEARD. *Maquettes et Prototypes.* Génie logiciel. n°3. Janvier 1986. pp.6.

[Lor87] F.J. LORENZ. *Reflections about Representation Methods. Workshop on the Future of Building Energy Modelling.* ISPRA. 1987.

[Mey88] B. MEYER. *Object-oriented Software Construction.* Prentice Hall International. 1988.

[MG75] MITCHEL - GAUTIER. *Advanced Continuous Simulation Language (ACSL)* . Concord. 1975.

[Mor81] T. MORAN. *The Command Language Grammar : a representation for the user interface of interactive computer systems.* International Journal of Man - Machine Studies. n°15. 1981. pp. 3-50.

[MD87] A.F. MONK - A. DIX. *Refininig early design decisions with a black box model.* People and Computers III. D. Diaper and R. Winder (Eds). Cambridge University Press. 1987.

[Min65] M.L. MINSKY. *Matter, Mind and Models.* Proc. IFIP Congress, Vol1. Spartan Books. pp.45-49. 1965.

[Mye88] B.A. MYERS. *A Taxonomy of Window Manger User Interfaces.* IEEE. Septembre 1988. pp. 65-84.

[Nak86] M. NAKHLE. *NEPTUNIX : Progiciel de Construction de Simulateurs Portables pour Processus Mixtes.* Note Applications Electroniques et Simulation. II-AES/02-86/MN.

[ND86] D.A. NORMAN - S.W. DRAPER. *User Centered System Design.* Lawrence Erlbaum Associates. Publishers. 1986.

[Ore78] T.I. OREN. *Concepts for Advanced Computer Assisted Modelling.* Methodology in Systems Modelling and Simulation. ed. North-Holland. 1978. pp. 29-55.

[PBD92] P. POYET - E. BRISSON - P. DEBRAS. *Environnements Logiciels pour l'Ingénierie Intégrée.* CSTB. DBC/92-1272.

[Poy90] P. POYET. *Integrated Access to Information Systems.* Applied Artificial Intelligence 4, 3. 1990. p. 179-238.

[Poy90] P. POYET. *Artificial Intelligence Software Engineering in Civil Engineering.* Microcomputers in Civil Engineering. 5, 3. 1990.

[PP79] A.B. PRITSKER - C.D. PEGDEN. *Introduction to simulation and SLAM.* Halsted Press New York. 1979.

[Pro81] PROBST. *Langages de Simulation.* Techniques et Sciences de l'Ingénieur. H2360. 1981.

[PS86] PHILIPON - SERMONDADAZ. *Modélisation et simulation à l'aide du processeur ALLAN.* Rencontres SFT. 1986. pp. 173 - 176.

[PS88] R. PELLETRET - S. SOUBRA. *CSTBât Jr - Un progiciel pour l'enseignement des techniques de Modélisation/ Simulation/ Analyse des résultats.* CSTB. DPE/88-574. 1988.

[RBG+86] J. N. J. RICHARDS - H.E. BEZ - D.T. GITTINS - D.J. COOKE. *On methods for interface specification and design.* International Journal of Man - Machine Studies. n°24. 1986. pp. 545-568.

[Rei81] P. REISNER. *Formal Grammar and Human factors in design of interactive graphics System.* IEEE Transactions on software Engineering. SE.7. 1981.

[Ril86] J. RILLING. *Modélisation et Simulation.* Cours de l'ENPC. 1986.

[Ron90] F.X. RONGERE. *CLIM 2000 basic Model Formalisation Method.* EDF.DER.ADE.HE12W2968. 1990.

[Sen87] B. SENACH. *Meta-communication, Gestion de contexte et Modélisation cognitive de l'utilisateur.* Bulletin de Liaison de la Recherche en Informatique et en Automatique. 115/1987. pp. 22-31.

[Sen90] B. SENACH. *Evaluation Ergonomique des Interfaces H/M : une revue de la litterature.* Rapport de Recherche. INRIA. 1990.

[SIK+82] D.C. SMITH - C. IRBY - R.KIMBALL - B. VERPLANK - E. HARSLEM. *Designing the Star User Interface.* Byte Magazine. Avril 1982. pp. 297-313.

[Sou88] S. SOUBRA. *CSTBât Jr pour Apple Macintosh - Manuel d'utilisation.* CSTB. TTA/DPE/88-678. 1988.

[SP89] S. SOUBRA - R. PELLETRET. *Interfaces Intelligentes pour les progiciels de modélisation / simulation.* Congrès Constructique 89. Atelier Formation Assistée par Ordinateur. TTA/DPE/89-733. 1989.

[SP89] S. SOUBRA - R. PELLETRET. *Le projet IISIBât.* Séminaire spécialisé "Outils d'aide à la conception et la gestion". AFME - Sophia Antipolis. CSTB. TTA/DPE/89-809. 1989.

[SP89] S. SOUBRA - R. PELLETRET. *Le projet IISIBât.* EuroplA90. Liège. 15-16 Mars 1990.

[SP91] S. SOUBRA - R. PELLETRET. *Intelligent Simulation Environments for Building Modelling and Simulation.* CIB W78. The Computer Integrated Future. Eindhoven. 16-17 Septembre 1991.

[SP91] S. SOUBRA - R. PELLETRET. *Intelligent Simulation Environments for Building Modelling and Simulation.* EuroplA91. Athènes. 26-27 Novembre 1991.

[Sow91] E.D SOWELL. *Next Generation Building Services Engineering Software : Opportunities for the Practitioner.* Building Environmental Performance 91. Canterbury. 10-11 Avril 1991.

[Str87] N.A. STREITZ. *Cognitive compatibility as central issue in Human/Computer interaction : theoretical framework and empirical findings.* Cognitive engineering in the design of human computer interaction and expert system. Amsterdam. Elsevier science publishers. 1987. pp. 75-82.

[SV82] SPRIET. VANSTEENKISTE. *Computer-Aided Modelling and Simulation.* Academic Press. 1982.

[Tam 91] S. TAMISIER. *Approche méthodologique de la modélisation pour la simulation des grands systèmes technologiques.* Thèse. Université d'Aix-Marseille III. 1991.

[TRN90] TRNSYS - *a TRaNsient SYstem Simulation program.* User-Manual. Solar Energy Laboratory. University of Wisconsin-Madison. 1990.

[Zei76] B.P. ZEIGLER. *Theory of modelling and simulation.* Wiley & sons New York. 1976.

GLOSSAIRE

Grandeur : Attribut d'un phénomène, d'un corps où d'une substance, qui est susceptible d'être distingué qualitativement et déterminé quantitativement. Les grandeurs qui sont mutuellement comparables peuvent être classées en catégories de grandeurs (par exemple : travail, chaleur, énergie).

Grandeur de base : pour des raisons en partie arbitraire, certaines grandeurs sont choisies comme grandeurs de base, considérées conventionellement comme indépendantes. Les autres grandeurs, appelées grandeurs dérivées, se déduisent des grandeurs de base par des relations entre grandeurs. L'ensemble des grandeurs de base, des grandeurs dérivées et des relations qui les lient forme un système de grandeur.

Dimension d'une grandeur : la dimension de la grandeur Q est l'expression A^{α} B^{β} C^{γ}... où A, B, C... indiquent les dimensions des grandeurs de base A, B, C... et où α, β, γ... sont appelés les exposants dimensionnels de la grandeur Q. Une grandeur dont tous les exposants dimensionnels sont nuls est dite grandeur sans dimension (cf. NF X 02-001).

Unité : grandeur déterminée, adoptée par convention, utilisée pour exprimer quantitativement des grandeurs de même dimension.

Unité de base : unité de mesure d'une grandeur de base dans un système donné de grandeurs.

Unité dérivée : unité de mesure d'une grandeur dérivée dans un système donné de grandeurs.

Système d'unités : ensemble d'unités établi pour un système de grandeur donnée. Un système d'unités comprend un ensemble d'unités de base choisies et d'unités dérivées déterminées par leur équations de définition et les facteurs de proportionnalité.

Par exemple : le système international d'unités SI est fondé sur les sept unités de base suivantes : mètre, kilogramme, seconde, ampère, kelvin, mole, candela. Les unités dérivées sont exprimés algébriquement en fonction des unités de base. (cf. NF X 02-006).

Modèle numérique : ensemble des équations algébro-différentielles implicites ou explicites, linéaires ou non, dont la résolution pour des valeurs initiales et/ou des sollicitations données, fournit une simulation du système. Cette résolution est menée en général par des moyens numériques du fait de la complexité des problèmes qui rend impossible la résolution formelle.

COMBINE : projet européen du programme JOULE de la DGXII des communautés européennes. Vise à développé un modèle commun de données permettant l'échange d'information entre divers outils d'évaluation, ainsi qu'entre des système d'aide à la conception.

IDM : "Integrated Data Model", tâche centrale du projet européen COMBINE.

MIPS : "Many Integrated Paradigm System" est un générateur d'applications intégrées multi-expertes, multimédia et hypertextes, développé au sein de la Division Bases de Connaissances du Service Informatique et Bâtiment du CSTB.

ANNEXE 1

FORMATS D'EXPRESSION DES DATES ET DES HEURES

Afin d'éviter toute interprétation erronée de la signification des chiffres désignant les dates et les heures, une utilisation de formats normalisés est nécessaire. La norme suivie est la norme ISO 8601 spécifiant la représentation numérique de l'information liée à la date et à l'heure du jour.

Dans le format d'expression des dates :

° le jour du mois (jour du calendrier) est représenté par deux chiffres. Le premier jour d'un mois quelconque est représenté par 01 et les jours suivants du même mois sont numérotés par ordre croissant ;

° le mois est représenté par deux chiffres. Janvier est représenté par 01 et les mois suivants sont numérotés par ordre croissant ;

° l'année est généralement représentée par quatre chiffres ; les années sont numérotées par ordre croissant à partir de l'an 0001.

La représentation complète est un élément numérique unique de huit chiffres représentant l'année, le mois et le jour.
Par exemple : 19850412.

Une date peut être exprimée avec une précision inférieure à celle indiquée ci-avant : on peut omettre deux, quatre ou six chiffres.
Par exemple : 198504
1985
19

On peut omettre les éléments de rang supérieur (troncature) quand leur présence est implicite pour l'application. Les éléments manquants sont alors remplacés par des tirets.
Par exemple : -8504
-85
--0412
--04
---12

Dans le format d'expression des heures du jour, les heures sont représentées par deux chiffres de 01 à 24, les minutes et les secondes étant représentées par deux chiffres de 01 à 60.

La représentation complète est un élément numérique comprenant six chiffres : hhmmss (hh figurant les heures, mm figurant les minutes et ss les secondes).
Par exemple : 232050

Si le degré de précision requis l'autorise, on peut omettre soit deux, soit quatre chiffre de la représentation complète.
Par exemple : 2320
23

Si une application particulière l'exige, on peut ajouter une fraction décimale de l'heure, de la minute ou de la seconde. Dans ce cas, on doit omettre les éléments de rang inférieur et séparer la fraction décimale de la partie entière à l'aide d'une virgule ou d'un point.
Par exemple : 232050,5
2320,9
23,3